TEXTURE REPRESENTATION AND ANALYSIS IN MATERIAL CLASSIFICATION AND CHARACTERIZATION

A Dissertation
Presented to
The Academic Faculty

By

Yuting Hu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2019

TEXTURE REPRESENTATION AND ANALYSIS IN MATERIAL CLASSIFICATION AND CHARACTERIZATION

Approved by:

Dr. Ghassan AlRegib
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sundaresan Jayaraman
School of Materials Science and Engineering
Georgia Institute of Technology

Dr. Ying Zhang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Surya Kalidindi
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: May 09, 2019

Nothing in life is to be feared, it is only to be understood.

Now is the time to understand more, so that we may fear less.

Marie Curie

I dedicate this thesis to,

My parents, **Bo Hu** and **Jinling Wang**,

Without the inspiration, drive, and support that you have given me, I might not be the person I am today.

My husband, **Zhen Wang**,

Thanks for your encouragement and support in my difficulties and for making my life so special and full of joy.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Prof. Ghassan AlRegib, for his guidance, support, and help throughout this research. During my first two years, Prof. AlRegib provided me every bit of guidance and assistance I needed. He taught me how to develop critical thinking skills, define research problems, conduct literature survey, make impressive presentations, and write high-quality research papers. After I was ready to conduct research by myself, he gave me enough freedom to work on topics I was interested in and kept contributing valuable feedback, suggestions, and encouragement. After five years of collaborations, he is not only my PhD advisor, but also one close friend in my life. Research is not a solo act, and collaborations accelerate the research progress and enhance the research quality. I would like to extend my sincere thanks to all former and current members of the Omni Lab for Intelligent Visual Engineering and Science (OLIVES) and the Center for Energy and Geo Processing (CEGP) for being my colleagues and friends. I am especially grateful to post-docs Zhiling Long and Can Temel as well as our lab members Motaz Al-Farraj, Chih-Yao Ma, Min-Hung Chen, Tariq Alshawi, Yazeed Alaudah, Gukyeong Kwon, Allie-Alexander Mussa, Muhammed Amir Shafiq, Mohit Prabhushankar, Charlie Lehman, and Jinsol Lee.

I would like to thank Prof. Anderson, Prof. Zhang, Prof. Sundaresan, and Prof. Kalidindi for serving on my Ph.D. dissertation committee. Their constructive comments and advice bring this dissertation to completion. In addition, I would like to extend my gratitude to all faculty in the Center of Signal and Information Processing (CSIP), especially Prof. Anderson for the substantial influence that his courses have had on my research. I also wish to thank Ms. Jane Chisholm, the instructor from the Center for Teaching and Learning (CETL), for her help on improving my academic writing skill. Without numerous instructions from her, my research papers would not have been published in good shape.

I would like to express my deepest appreciation to my family for their love and support

through the Ph.D. journey. I would like to especially thank my husband, without whose adequate understanding, constant encouragement, and great sacrifice the completion of the journey would not have been possible. Many thanks go to my friends for your company in this challenging but enjoyable journey. Finally, I wish to acknowledge all who assisted me directly or indirectly in past years. Because of you, five years at Georgia Tech will be the most memorable experience in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Objects and scenes in the real world exhibit abundant textural information. Textures observed in the surfaces of natural objects do not only capture the appearance of materials, but also contain important visual cues that are perceived by the human visual system. By perceiving and understanding textures, humans learn about objects and scenes in more detail and interact better with the physical world. In a similar fashion, when dealing with real-world images, computers need to extract information from the pixels of textures and understand represented patterns. Texture representation, as the core of texture analysis and image understanding, aims to extract descriptive features that provide useful information in identifying local regions or the characteristic properties of texture patterns. Since textures exhibited on object surfaces reveal material types and properties, an effective texture representation algorithm can be helpful for analyzing and understanding associated materials. In recent years, recognition and characterization of textures and materials from their visual appearance has been effective in various applications including object recognition [1, 2], robotic manipulation and grasping [3, 4], quality inspection and assessment [5, 6, 7], scene understanding [8], facial image analysis, medical image analysis [9], and geological structure interpretation [10].

In the industry, modern automated manufacturing systems utilize visual information to perform specific tasks such as computationally efficient visual tracking and surface characterization of products. Visual tracking and surface characterization, however, are still in the preliminary stages of research and are currently underdeveloped. However, these tasks are commonly conducted on products at the macroscopic scale. At this level, the materials have simple structures. When dealing with products at the microscopic scale, manufacturing tasks face more challenges in the decision making process that mitigate the ability of a computational system to perform the tasks. Therefore, many manufacturing tasks conducted on products at the fine-grained scale are still subjectively performed by experts

who manually interact with products based on their specialized domain knowledge. The drawbacks of such subjective processes include extensive labor, considerable time, human errors, and inconsistency in evaluation. It is desirable to develop an automated system that can perform manufacturing tasks at both macroscopic and microscopic levels in order to minimize these drawbacks.

Although some manufacturing system methodologies [11, 12, 13] have been proposed to improve the visual tracking and surface characterization of products at fine-grained levels, they fail to describe objects or materials in a discriminative and efficient manner which limits their practical applications. To remedy this shortcoming and improve the manufacturing system, texture can be involved. By capturing an image of the surface of a manufactured product, a modern intelligent or automated manufacturing system can utilize the textural information presented in the image to recognize and track specific texture patterns, characterize surface features, and provide a feedback into the manufacturing cycle on indicators such as production quality. However, because of limited available data, texture recognition as a more general problem of material recognition has not been fully investigated. Recently, recognition and characterization of textures have attracted more research attention due to the increased availability of texture or material datasets.

Current research directions of texture recognition mainly fall into two categories. One category is the recognition or identification of classes of various textures for images captured in an unconstrained environment, hereon referred to as "in the wild". Those images captured in real-world scenarios may contain a wide range of generic texture categories and be occupied partially or entirely by one or several types of textures. The other category is the characterization of material properties "at the fine-grained scale" for images generated in controlled lab environments. Generally, such an image is entirely occupied by only one type of textures in such controlled environments. Differentiating texture images from similar products with different surface characteristics is challenging because of their similar visual appearance. Therefore, more discriminative features should be extracted to

distinguish subtle details.

Texture analysis algorithms involving discriminative texture descriptors promote the development of automated manufacturing systems. Developed texture descriptors are utilized to mainly solve three main vision-based tasks, motion tracking of textures, texture or material recognition, and surface characterization. The objective of this thesis is to design discriminative texture descriptors by developing handcrafted and deep learning-based texture representation methods to aid in the deployment of such automated manufacturing systems.

Motion Tracking of Textures: Binary descriptors have two important advantages on local texture representation. One is their low computational cost, and the other is that the binary features of interest keypoints between video frames can be efficiently matched. Meanwhile, non-binary texture descriptors perform discriminative and stable representation ability. Combining these two types of descriptors achieves a trade-off between discriminative representation and efficient computation. To recognize texture patterns and track the position of materials in a video, we develop a discriminative and efficient texture representation method by combining binary and non-binary handcrafted local feature extraction approaches. In addition to utilizing the information of local texture appearances, we apply global geometric constraints of texture patterns underlying materials for accurate and efficient texture tracking.

Material Classification based on Texture: Describing and distinguishing materials "in the wild" with major intra-class textural variations and "at the fine-grained scale" with minor inter-class textural variations between similar products with different surface characteristics are challenging. To overcome this problem, we design texture representation algorithms in a progressive manner evolving from deterministic to learning-based ones. We develop binary descriptors, completed local derivative patterns (CLDP) and scale selective extended local binary patterns (SSELBP), the discriminative ability of which is improved by mutli-scale information. In addition, we introduce a more discriminative

non-binary texture representation method, block intensity and gradient difference (BIGD), which maintains intensity differences between a pair of image blocks in multiple orientations and scales. Furthermore, we develop an end-to-end multi-level texture encoding and representation network on the basis of convolutional neural network (CNN) features, texture encoding layers, and multi-scale analysis. These learning-based features do not only show superior performance on material classification, but can also potentially be applied to other visual tasks.

Material Surface Characterization: Characterizing properties of surface images (i.e., computational surface characterization) poses a challenging problem when surface characteristics of interest may be more latent than apparent in the sense that the characteristics are not always easily differentiated in the visual appearance. To deal with this problem, we develop a multi-level texture encoding and representation network (MuLTER) that integrates both low-level and high-level CNN features to achieve a texture representation which maintains both texture details and local spatial information. Additionally, we introduce a publicly available dataset for surface characterization in which we assess our algorithm and existing deep learning-based texture analysis techniques for material surface characterization. In summary, our dataset and algorithm will serve as a benchmark evaluating deep learning-based techniques for both material surface characterization and more general texture classification.

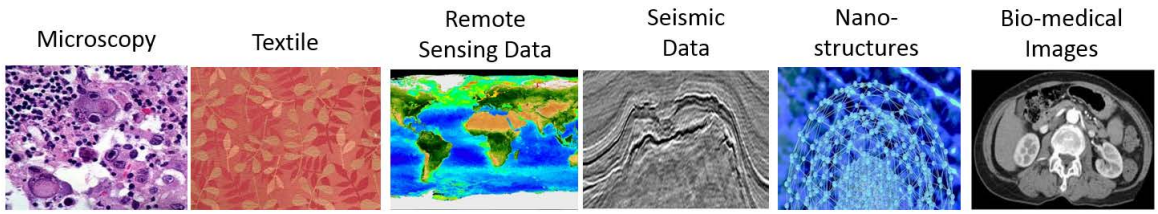
CHAPTER 1
INTRODUCTION AND BACKGROUND

Our world is filled with diverse and complex textures from various real sources. Such texture can be captured for analysis by various different imaging techniques. As such, textures exhibit differing visual appearances in differing image types such as object surfaces in natural images [1, 2], fabric patterns in microscopic images [6], and geological structures in seismic images [10]. Fig. 1.1 illustrates examples of textures in natural images and other specific types of images. Visual texture information presents the viewer with the feel or appearance of an object, which contains latent information on several key characteristics of the material such as smoothness, regularity, or granularity, which are commonly determined by the size, shape, and arrangement of the corresponding material micro-structures. By representing the spatial organization of the basic elements of the objects that compose an image or video, the fundamental micro-structures of a materials textures provide important visual cues for our preattentive human visual system to identify objects or local regions.

When processing the textural information for interpretation, however, the appearance of textures presents some degree of variability ranging from perfectly regular (e.g., periodically repeated textons) to randomly irregular. These variations in appearance lead to two different types of perceptual properties of textures: *describable* or *non-describable* textures. For example, a *describable* texture can be described by universal semantic texture characteristics [14] such as "regular, sparse, and dotted" or "noisy, line-like, and irregular" [15]. These descriptions elicit the rich visual impressions a surface or substance makes on the viewer prior to high-level semantic visual understanding in human perception. In contrast, *non-describable* textures are characterized abstractly with local and global statistical features of the image.

The representation and analysis of textures open the door to several diverse and appeal-

(a) Textures in natural images



(b) Textures in special scenarios

Figure 1.1: An example of the various textures found in natural images (a) and various other image acquisition domains (b).

ing applications. Some recent works that utilized the representation and analysis of texture are in areas such as object recognition [1, 2], robotics/autonomous navigation [16, 3, 4], quality inspection and assessment [5, 6, 7], scene understanding [8], facial image analysis, image and video editing [17], crowd behavior analysis [18], remote sensing [19, 20], geological structure interpretation [10], and medical image analysis [9]. From these literary works, it is noted that texture analysis typically covers four main problems: classification, segmentation, shape from texture [21], and synthesis. These four problems are closely related and share the same core technique, as their successful completion relies on accurate texture representation.

In this thesis, we focus on developing novel texture representation methods and investigating their roles in three main vision-based tasks: the detection and tracking of texture patterns, the classification or recognition of texture/materials, and the automatic assessment of material surface characteristics of interest.

1.1 Texture Representation

Images of real-world objects and scenes exhibit an abundant amount of textural information. Appropriately describing these textures is an essential and challenging problem in the

field of computer vision and pattern recognition. Developing a computational understanding of those images relies heavily on the methodologies used to provide a description of the textures presented to the algorithm. This description of the texture for computational analysis is referred to as texture representation.

Texture representation, the core of texture analysis, extracts discriminative features that describe texture information from images and has attracted attention from both academia and industry in recent years. One main challenge in texture representation is the balance between discriminative representation and computational efficiency. In real-world images and videos, a wide range of texture categories may present apparently large intra-class visual variations due to diverse photometric and geometric conditions. In contrast, textures in different subcategories may exhibit non-discernable inter-class visual variations. Therefore, to deal with such large intra-class and small inter-class variations in the visual perception of the textures, the chosen texture representation needs to strongly discriminate against the features extracted from the image. However, this leads to an increase in computational complexity of the representation method. Increasing the computational complexity of the texture representation methods constrains their practical applications, especially in circumstances of limited computing resources.

Researchers have proposed various approaches for texture feature extraction that aim to solve this conflict that arises from these two competing goals, such as predefined filter banks [22, 23], statistical models [24, 25], and local texture descriptors [26, 27, 28, 29, 30, 31, 2]. The representative features extracted by the approaches in the literature are derived from conventional texture datasets. They are effective in scenarios with insufficient annotated training data or limited computing power. Since these features are derived in a deterministic manner from the information presented in the image, they are known as hand-crafted features.

Nowadays, with the rapidly growing amount of image and video data, hand-crafted features cannot capture all of the available texture details presented in an image. With

the recent emergence of deep learning techniques, convolutional neural networks (CNNs) have achieved record-breaking accuracy on image classification tasks. The CNN serves as an efficient feature extractor and can learn more generic information compared to that of hand-crafted features from large-scale annotated datasets (e.g., ImageNet [32]). Although CNNs exhibit this desirable generalized information representation, CNN-based approaches trained on natural image datasets mainly capture the features of the real-world objects rather than the textures.

To obtain texture features with rich discrimination power from such features of real-world objects, features learned from natural images need to be carefully transferred to the texture domain. The state-of-the-art methods of CNN-based texture representation focus mainly on two perspectives: the combination of pre-trained CNN features and feature encoding [15, 14, 8] and end-to-end feature learning networks with texture encoding layers [33]. In the work of [15], Cimpoi et al. proposed a Fisher-vector based CNN (FV-CNN), which calculates Fisher vector encoding [34] on top of a CNN pretrained on the ImageNet dataset. One shortcoming of the FV-CNN method is that the modules of feature extraction, texture encoding, and classifier training are isolated, which fails to utilize all of the benefits from having a large-scale labeled dataset. To jointly learn all of these modules, an end-to-end learning method [33] has been proposed recently.

In this thesis, to generate more discriminative features, we will introduce various texture representation methods ranging from hand-crafted to CNN-based.

1.2 Vision-based Tasks

1.2.1 Motion Tracking of Textures

Currently, research on texture analysis has facilitated the rapid growth of robotics and industrial inspection. One such industrial application that has benefited from the advancements of texture analysis is the garment manufacturing industry, one of the oldest and largest industries in the world. Different from other mass industries such as the automobile

industry, the apparel industry is primarily supported by a manual production line with little to no automation of processes.

Since textile fabrics are easily sheared or distorted, the performance of motion and quality control of manufactured fabrics is generally the most critical process and traditionally relies heavily on manual labor. For materials with regular textures, an intelligent vision system is needed to perform a series of visual tasks for motion and quality control: (1) extracting representative texture features and distinguishing various material types, (2) recognizing near-regular texture patterns presented in fabrics, and (3) tracking the position of fabrics. To solve these problems, we will design a system that utilizes texture features to implement the detection and tracking of texture patterns.

1.2.2 Texture and Material Recognition

The discriminative ability of texture representation methods can be evaluated by their performance on texture classification tasks. With the increasing crowd-source annotation of texture datasets on the Internet, texture classification (i.e., texture/material recognition) has become a popular topic among the community of machine learning and computer vision. The main task of texture classification is to differentiate the data in a dataset into texture categories or instances. Current research directions for texture classification present two trends: (1) recognizing categories of texture or material images captured "in the wild", under challenging environments or even in the real world, and (2) differentiating texture or material subcategories details "at the fine-grained scale", typically in an ideal microscopic environment. In practical applications, texture classification is a challenging problem since categories "in the wild" may have large intra-class variations and subcategories "at the find-grained scale" may have small inter-class variations.

Typically, textures observed in natural scenes are representative of various materials. Therefore, successful texture representation can be helpful for analyzing and understanding the associated materials. Recently, automated material analysis has attracted increasing

interests from academia and industry because of its potential applications, spanning from scene understanding to robotics and industrial inspection. For example, if an autonomous vehicle knows what kind of ground terrain it is driving on (i.e., whether the ground surface material is concrete, asphalt, soil, or pebbles), it can adjust itself according to the actual outdoor environment to ensure a safe operation [35].

Existing research on automated material analysis mainly focus on material recognition, the task of which is to classify various types of materials into their associated categories. Typically, such classification is coarse-grained, which means that materials of interest cover a wide range of generic categories. For example, in [36], material categories involved in experiments include fabric, foliage, glass, leather, metal, paper, plastic, stone, water, and wood. In [35, 16], the materials ground surfaces are classified into categories such as asphalt, concrete, pebbles, and soil. In some cases, fine-grained material classification has been studied, where the subject materials belong to the same generic category but different subcategories. As an example, in [6], the generic category of fabric is divided into subcategories such as cotton, terrycloth, denim, fleece, nylon, polyester, silk, viscose, and wool.

1.2.3 Material Surface Characterization

Although material recognition is important, we believe material analysis should also address the topic of material characterization. This is an analysis that takes a step further beyond merely recognizing the material. It aims to find out more detailed information about a material in terms of certain specific properties of the material. For example, for a robotic arm to catch a glass container, not only does it need to recognize the material being glass (thus being fragile), but it also has to know further the level of fragility of that particular glass container, so that it can apply appropriate level of pressure to the object when catching it. Such problems have rarely been studied in the image processing and computer vision literature, but are also of practical significance. Essentially, material char-

acterization can be considered as fine-grained classification, where the categories belong to the same material (e.g., glass) but represent different levels in terms of a certain property of the material (e.g., fragility).

Despite the recognition and detection of texture patterns mentioned above, we study material characterization in the context of intelligent manufacturing systems in the textile industry, where an automatic surface characterization system is needed to perform an objective texture attribute assessment of manufactured fabric surfaces. Traditionally, the surface characterization is performed by a subject expert who manually touches the fabric. Not only does such a subjective assessment demand skills and experience, but it also has major drawbacks including extensive labor, consuming considerable amount of time, and most undesirably, suffering from possible human errors and inconsistency. Consequently, it is desirable to develop an automated system that can perform a material surface characterization instead. Such systems can provide the benefits of reliability, consistency, and efficiency to that of the manual process.

The primary objective of such an automated system is to examine material surfaces in terms of the characteristics of interest such as the relative fiber length and smoothness on the fabric surface. Few works in the literature (e.g. [37]) have attempted to characterize fabrics via objective means using chemical, physical, or mechanical measurements with limited success. We believe texture-based image analysis is a feasible approach here, as the target characteristics are visual and tactile properties observed at fabric surfaces which have been demonstrated to be generally correlated with visual properties of texture images [38, 39]. Some material characteristics of interest may be more latent than apparent, not always easily differentiated in the visual appearance. Therefore, advanced techniques such as deep learning-based texture analysis is indispensable for the material characterization task.

The remainder of this thesis is organized as follows. Chapter 2 conducts a literature survey of texture representation and analysis and summarizes literature related to texture tracking and pattern detection, texture/material classification, and material surface char-

acterization. Chapter 3 discusses novel binary texture representation methods and their roles in texture tracking and classification. Chapter 4 presents a discriminative non-binary texture representation using gradient-based features and multi-scale analysis. Chapter 5 introduces a deep learning-based texture representation method and applies it to material recognition and material surface characterization. Chapter 6 makes a conclusion for the entire dissertation.

CHAPTER 2
LITERATURE SURVEY

2.1 Texture Representation and Analysis

The purpose of texture representation and analysis is to extract powerful features that provide important visual cues or characteristic properties of objects and materials, including fundamental micro-structures (i.e., texture elements, texture primitives, or textons), spatial arrangements, and visual or tactile properties. In this chapter, we introduce the development and challenges of texture representation in Sec. 2.1.1 from traditional representation techniques to recent machine/deep learning based approaches. In Sec. 2.1.2, we discuss a standard pipeline for texture representation and analysis. In addition, we investigate popular deep learning based methods in Sec. 2.1.3. As different texture representation approaches are adapted for various scenarios, we focus on developing texture representation methods for three vision-based tasks: texture tracking, texture/material classification, and material surface characterization, respectively. The related work is summarized in Sec. 2.2.

2.1.1 The Development and Challenges of Texture Representation

Over the years, the image processing community has worked on diverse texture analysis problems including texture perception [38], texture description [40, 14], material recognition [22], texture segmentation [23, 14], and texture synthesis [41]. Earlier work about texture analysis dated back to the 1970s when Haralick et al. [42] proposed a texture descriptor, the gray level co-occurrence matrix (GLCM), which extracts co-occurrence grayscale values and collects their statistics. In the early 1980s, Julesz [43] proposed the concept of "texton", which stresses the importance of micro-structures (i.e., edges, corners, and blobs) for pre-attentive human perception and texture discrimination. Since the 2000s, Leung and

Malik [22] introduced "predefined" filter banks with various scales and orientations to local regions and utilized the distribution of local filter responses to characterize textural information.

To mathematically establish the model of textures, the bag of textons (BOT) [22] and later the bag-of-words (BOW) are designed, which learn from local texture descriptors a dictionary of visual key words and generate a histogram-based distribution of local feature vectors based on the dictionary. Representation based on the BOW model has become a popular module in texture or object recognition and image understanding. BOW combined with local descriptors such as the scale-invariant feature transform (SIFT) [27] or local binary patterns (LBP) [44] had been the most widely used texture representation method. Nowadays, the development of machine learning has shed a light on texture representation and analysis. Since 2012, when deep CNN [32] marked a milestone on image recognition, texture analysis via deep learning methods has been explored in [8, 14, 15]. In summary, the directions of texture representation methods mainly include two main categories, hand-crafted approaches and deep learning-based methods, each of which we will discuss in Sec. 2.1.2 and Sec. 2.1.3, respectively.

Similar to ImageNet [52] for object recognition and natural scene classification, large-scale texture and material image datasets have been created through both Internet-based crowd-source annotation and in-lab controlled acquisition in recent years. We provide a summary of popular texture datasets in Table 2.1 and compare them from different perspectives such as the number of total images, image size, imaging environments, image content, image variations, etc. Among these datasets, the Flickr material database (FMD) [36] and describable textures dataset (DTD) [15] are moderate-sized datasets for recognizing describable texture attributes in natural images. The materials in context database (MINC) [50] is a large-scale, diverse, and well-sampled dataset, with 23 categories and three million material samples. By including Flickr images and Houzz photos from Internet, the MINC dataset represents a wide range of materials in the real world, facilitating

Table 2.1: Comparison between CoMMonS dataset and some publicly available texture and material datasets. The comparison follows the example in [45] and reuses its relevant information.

Name	Total Images	Classes	Image Size	Gray/ Color	Imaging Environment	Illumination Changes	Rotation Changes	Viewpoint Changes	Scale Changes	Image Content	Scenes	Grain Level	Year
CUReT [46]	5612	92	200×200	Color	Constrained	Yes	Small	Yes	No	Materials	No	Coarse	1999
KTHTIPS [47]	810	10	200×200	Color	Constrained	Yes	Small	Small	Yes	Materials	No	Coarse	2004
FMD [36]	1000	10	512×384	Color	Wild	Yes	Yes	No	No	Materials	In scene	Coarse	2009
OpenSurfaces [48]	10422	22	Unfixed	Color	Wild	Yes	Yes	Yes	Yes	Materials	In scene	Coarse	2013
DTD [15]	5640	47	Unfixed	Color	Wild	Yes	Yes	No	Yes	Textures	In scene	Fine	2014
TUM Haptic Texture [49]	690	69	224× 224	Color	Controlled	Yes	Yes	No	Yes	Materials	No	Fine	2014
MINC [50]	2996674	23	Unfixed	Color	Wild	Yes	Yes	Yes	Yes	Materials	In scene	Coarse	2015
MINC2500 [50]	57500	23	362×362	Color	Wild	Yes	Yes	Yes	Yes	Materials	In scene	Coarse	2015
4D Light-field [51]	30000	12	128× 128	Color	Constrained	No	No	No	No	Materials	In scene	Coarse	2016
IC-CERTH Fabric [6]	1266	9	640×480	Color	Constrained	Yes	No	No	No	Materials	No	Fine	2016
GTOS [35]	34243	40	240×240	Color	Partially Constrained	Yes	Yes	Yes	No	Materials	Scene	Coarse	2016
GTOS-mobile [16]	34243	31	240×240	Color	Partially Constrained	Yes	Yes	Yes	No	Materials	Scene	Coarse	2018
CoMMonS (ours)	6912	12	1920×2560	Color	Constrained	Yes	Yes	No	Yes	Materials	No	Fine	2019

material recognition in unconstrained conditions (i.e., in the wild). Most recently, a large-scale material surface dataset, ground terrain in outdoor scenes (GTOS), with over 30,000 images covering 40 classes was collected by Xue et al. [35], geared towards real-world material recognition for autonomous agents. These datasets were useful to evaluate texture representation and analysis methods. However, as the texture datasets discussed above were all created for material recognition, they are not suitable for material characterization where fine-grained classification is performed within one type of material in terms of some certain material characteristic or property.

Texture representation is particularly challenging in two aspects. In a coarse-grained classification problem, intra-class variations can sometimes be significant because of diverse photometric and geometric conditions. In contrast, in a fine-grained texture classification problem, inter-class appearance variations are smaller and are not always easily distinguishable. Better understanding of textures with various variations requires discriminative, robust, and efficient texture representation methods. "Discriminativeness" determines that texture descriptors differentiate texture patterns captured under various geometric and photometric conditions; "efficiency" indicates that the implementation complexity and feature dimensions of texture descriptors are satisfactory for real-time purposes or limited computing power; and "robustness" denotes that texture descriptors deal with various imaging distortions such as noise and deformation. Since these competing goals have not been completely solved by existing approaches, advanced techniques such as deep learning-based texture analysis are indispensable for various texture representation and analysis related vision tasks.

2.1.2 Texture Representation and Analysis Pipeline

We show a general pipeline of texture analysis in Fig. 2.1 and its goal is to transform an image patch or an entire image into a compact feature vector that describes texture structures or properties. The extracted feature vectors are then adapted for visual tasks

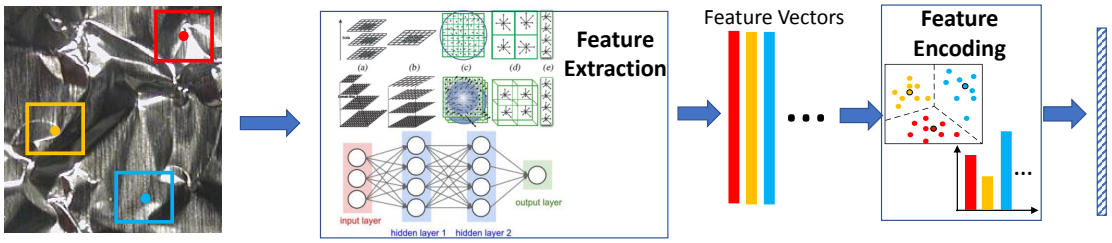


Figure 2.1: Texture representation and analysis pipeline.

under different scenarios. The transformation from image pixels to feature vectors usually involve two major steps: feature extraction and feature encoding, both of which we will discuss next.

Local feature extraction

Local texture descriptors [53, 67], which are robust against rotations and translations of images, are able to provide discriminative features for describing local image regions. Previous research efforts have focused on extracting local descriptors in a discriminative and efficient way. We summarize popular local feature extraction methods in Table 2.2(a) and Table 2.2(d). The existing approaches are commonly divided into four categories including covariance-, filter-, gradient-, and binary-based descriptors. Covariance-based descriptors modeling the second-order statistics of images perform well on material categorization [59]. However, by retaining only the second-order statistics, covariance descriptors are prone to be singular and have limited capability in modeling nonlinear, complicated feature relationships. Filter-based descriptors [28] acquire local features using filter banks. One problem of filter-based descriptors is that the design of filter banks is data dependent. Among gradient-based descriptors, SIFT [14] and speeded up robust features (SURF) [53] as the most popular ones capture the discriminative gradient features of local image patches.

Binary descriptors (e.g., LBP [67], binary robust invariant scalable keypoints (BRISK) [54], and local binary difference (LDB) [68]), which convert the intensity differences of neighboring pixels to binary values, are robust to monotonic illumination changes and re-

Local Feature Extraction	Year	Key Features \ Approaches
Lowe et al. [27] (SIFT)	2004	Scalable invariant feature transform
Bay et al. [53] (SURF)	2006	Speeded up robust features
Leutenegger et al. [54] (BRISK)	2011	Binary robust invariant scalable keypoints

(a) Feature extraction

Feature Encoding	Year	Key Features \ Approaches
Varma & Zisserman [28] (BoW)	2005	Bag of visual words
Perronnin & Dance [34] (FV)	2007	Fisher vectors
Jégou et al. [55] (VLAD)	2010	Vector of locally-aggregated descriptors

(b) Feature encoding

Texture Tracking	Year	Key Features \ Approaches
Book et al. [11]	2010	2D FFT and the Harris corner detection
Hays et al. [56]	2010	Higher-order correspondence, optimization, and the Harris corner detection
Lin & Liu [57]	2006	Markov random field (MRF)
Hu et al. [39]	2018	Texture pattern extraction with geometric constraints

(c) Texture tracking and lattice detection

Texture Classification	Year	Key Features \ Approaches
Varma & Zisserman [58]	2009	Filter banks
Harandi et al. [59]	2014	Covariance descriptors
Ojala et al. [44] (LBP)	1996	Traditional local binary patterns
Hu et al. [60] (CLDP)	2016	Completed local derivative patterns
Liu et al. [61] (MRELBP)	2015	Extended LBPs
Guo et al. [62] (SSLBP)	2016	Scale selective LBPs
Hu et al. [63] (CLDP)	2016	Scale selective ELBPs
Mehta & Eguiazarian [64] (DMD)	2016	Dense micro-block difference
Hu et al. [65] (BIGD)	2019	Block intensity and gradient difference
Cimpoi et al. [14] (FV-CNN)	2015	DeCAF and fisher vector (FV) encoding
Zhang et al. [33] (DEEP-TEN)	2017	Deep texture encoding networks
Xue et al. [16] (DEP)	2018	Deep encoding pooling networks

(d) Texture classification

Surface Characterization	Year	Key Features \ Approaches
Tamura et al. [38]	1978	Textural features corresponding to visual perception
Ferrari & Zisserman [40]	2008	Visual attribute learning
Cimpoi et al. [14]	2015	Texture attributes and semantics
Kampouris et al. [6]	2016	Fine-grained classification with micro-geometry
Sun et al. [7]	2018	CNN-based automatic surface roughness estimation
Hu et al. [66]	2019	Deep learning-based texture attribute assessment

(e) Material surface characterization

Table 2.2: An overview of existing feature extraction, feature encoding, texture tracking, texture/material classification, and material surface characterization.

quire low computational cost. These advantages make binary descriptors appealing for real-time applications. We summarize commonly used binary descriptors in Table 2.3. Since texture patterns show different visual appearances caused various geometric and photometric image capture conditions, the categorization of texture types with various variations into the same group is an important factor during the design of binary descriptors.

Table 2.3: An overview of LBP variants.

Method	Compact Encoding	Rotation Invariance	Neighborhood Sampling	Scale Invariance	Uniform Patterns	Multi-scale Analysis	Year
LBP [44]			✓				1996
"ri"-LBP [26]		✓	✓				2000
Multi-scale "riu"-LBP [69]		✓	✓		✓	✓	2002
Dominant LBP (DLBP) [70]		✓	✓			✓	2009
Completed LBP (CLBP) [69]	✓	✓	✓		✓	✓	2010
Local derivative pattern (LDP) [71]			✓		✓	✓	2010
Extended LBP (ELBP) [72]	✓	✓	✓		✓	✓	2010
Local directional derivative pattern (LDDP) [73]		✓	✓		✓	✓	2012
Median robust ELBP (MRELBP) [74]	✓	✓	✓		✓	✓	2015
Scale-selective LBP (SSLBP) [62]		✓	✓	✓	✓	✓	2016
Completed local derivative pattern (CLDP) [60]	✓	✓	✓		✓		2016
Scale-selective ELBP (SSELBP) [63]	✓	✓	✓	✓	✓	✓	2017

Aiming at solving the issue of illumination variations, Ojala et al. [44] proposed LBP, which describes the sign information of local intensity differences between each pixel and its neighboring pixels. By encoding the sign information in a circular manner into binary codes, LBP can achieve high efficiency on texture representation. Because of LBP's great success in face recognition and texture classification, several methods derived from LBP have been introduced in recent years. Guo et al. [69] proposed CLBP, which extracts not only local sign and magnitude information, but also global intensity information. Without uniform patterns in LBP, Liao et al. [70] proposed the dominant LBP (DLBP), which computes the occurrence frequencies of all rotation invariant LBP patterns and defines the most frequent ones as dominant patterns. Moreover, to enhance classification performance and robustness, LBP-based algorithms [69, 75, 73, 76, 77] involving multi-scale patterns were proposed. However, almost all of these LBP-based methods ignore local derivatives, which contain complementary discriminative information [71]. To address this drawback, Zhang et al. [71] proposed the local derivative pattern (LDP) to encode the local derivative information of various directions. Although LDP shows good performance on face recognition,

it cannot ensure rotation invariance in texture classification without a circular coding strategy. To accomplish rotation invariance in local derivative patterns, Guo et al. [73] proposed the local directional derivative pattern (LDDP). However, the performance of LDDP is not comparable to that of CLBP in texture classification because of the lack of magnitude information. These LBP variants are robust to gray-scale and rotation variations, but suffer from scale variations.

For the purpose of scale invariance, the work of [78] introduced a feature extraction method that implements scale-invariance by estimating local scales and normalizing local regions, but has high computational complexity. To improve efficiency, Guo et al. [62] proposed the scale-selective LBP (SSLBP) that first extracts scale-sensitive local features and then applies a global operator to achieve scale-invariance. In [62], the scale-invariant feature extraction scheme achieves good texture classification performance on texture databases with scale variations such as KTH-TIPS [79] and UMD [25]. However, SSLBP as a high-dimensional descriptor has a length of 2400. To reduce the feature dimension, Liu et al. [72] proposed ELBP, which well represents texture images and achieves good texture classification performance using limited features.

The major drawback of binary descriptors is that the binarization of local intensity differences leads to the loss of intensity information, which weakens the ability of discrimination. Another disadvantage of binary descriptors is their dimensions, which will grow exponentially when the number of pairwise comparisons on neighboring pixels increases. To alleviate these problems, non-binarized texture descriptors were proposed. Zhang et al. [80] proposed a descriptor named normalized difference vector (NDV) and Mehta et al. [64] proposed a novel descriptor called dense micro-block difference (DMD). Both methods composed of real-valued intensity differences instead of binary codes of different micro-blocks in local patches. Although DMD captures non-quantized patch-based features at multiple scales and orientations, the neglect of first-order gradients may deteriorate the discriminative ability [53][68]. Therefore, for improving the discriminitave ability

of texture descriptors, more gradient features can be involved.

Feature encoding

The objective of this step is to learn texton dictionary (i.e., codebook) and link local texture representation and feature pooling as a global feature representation of an image. We summarize commonly used feature encoding methods in Table 2.2(b). Feature encoding or feature pooling in the BOW pipeline first uses the K-means algorithm and obtains K clusters (i.e., a dictionary of visual words) of individual local descriptors from training samples. Then, by assigning each local descriptor to its nearest visual word (i.e., a hard assignment), the BOW encoder calculates a histogram of visual word occurrences.

To include richer information instead of simple occurrences, two popular extensions of BOW are Fisher vectors (FV) [34] and vector of locally-aggregated descriptors (VLAD) [55]. Different from BOW, FV replaces K-means clustering with Gaussian mixture models (GMM). FV also replaces hard assignment with soft assignment from the posterior probability of each GMM. In addition, FV encodes both first- and second-order statistics of descriptors. VLAD is a variant of FV. Similar to BOW, VLAD first assigns individual local descriptors to their nearest visual words. Then, for each visual word, VLAD accumulates the differences between this visual word and its corresponding local descriptors. Different from BOW, VLAD aggregates the first-order statistics of descriptors.

2.1.3 Deep Learning-based Texture Representation

Deep learning-based methods have been explored for texture representation and analysis among the society. Texture/material recognition generally is challenging in demanding an orderless representation of micro-structures (i.e., texture encoding). Previous research generally combines concatenated global CNN activations and a fully connected layer (as a classifier), which fails to meet the need for a geometry-invariant representation describing local feature distributions. To overcome this drawback, Cimpoi et al. [14] first proposed

a Fisher-vector CNN descriptor (FV-CNN), which significantly boosts performance for texture analysis-related vision tasks. First step is to extract generic deep convolutional activation features called DeCAF. A deep CNN is pre-trained on ImageNet [32] and most experiments build on AlexNet [32] and VGG-M models [81]. By removing the softmax and last fully-connected layer of the network, DeCAF is represented by a feature vector. Second step of FV-CNN is to compute FV on DeCAF features. The FV pooling of DeCAF is the best so far, but its computational complexity is high. In addition to FV-CNN, a survey paper [82] evaluates other data-driven approaches including ScatNet and PCANet. To evaluate handcrafted texture descriptors and deep learning-based counterparts, [82, 14] provide a systematic review and test their discriminativeness, robustness, and computational efficiency on typical texture and material datasets.

One shortcoming of the FV-CNN architecture is the separation between CNN feature extraction, texture encoding, and classifier training, which does not benefit from the labeled data. To jointly learn them together in an end-to-end manner, Zhang et al. [33] proposed a texture encoding layer, which builds the dictionary learning and feature encoding on top of the CNN architecture. This deep texture encoding network (Deep-TEN) learns an orderless image representation, which performs well on texture or material recognition. But as textures do not always exhibit completely orderless patterns, local spatial information is still useful for differentiating them. To address this issue, Xue et al. [16] presented a deep encoding pooling network (DEP), which fuses orderless texture encoding and local spatial information to yield enhanced performance over Deep-TEN. However, neither Deep-TEN nor DEP fully utilizes CNN features from different layers and resolutions, leaving room for further development.

2.2 Texture Tracking, Texture and Material Recognition, and Material Surface Characterization

2.2.1 Motion Tracking of Textures

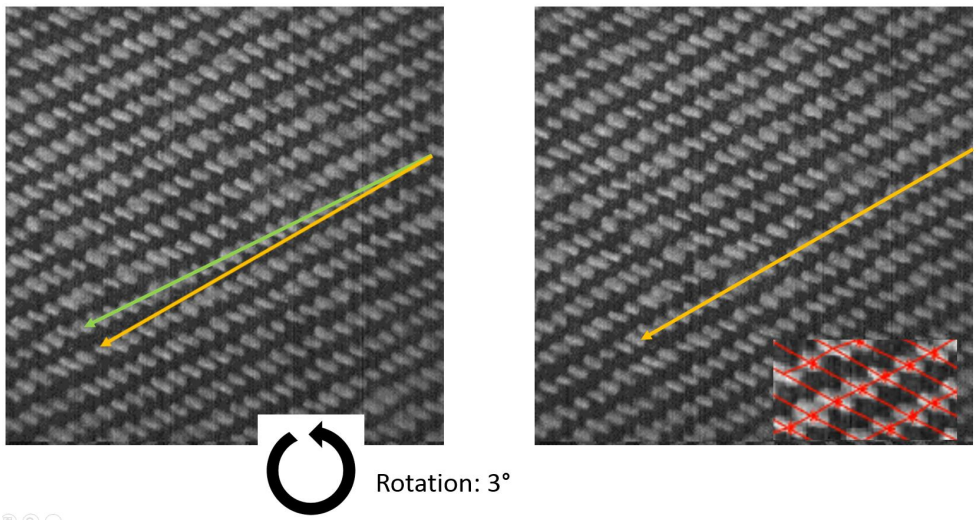


Figure 2.2: An illustration for texture tracking.

Real-world examples of texture patterns such as fabric patterns or architecture surfaces are numerous, especially in man-made environments. Detecting texture patterns and tracking locations/motions of manufactured products in Fig. 2.2 in an effective and efficient manner are important for visual tracking. We summarize the literature for texture tracking in Table 2.2(c). Hays et al. [56] and Lin and Liu [57] came up with algorithms for accurate texture tracking. However, the optimization and iteration process in their algorithms are complex and time consuming, not suitable for real-time vision applications. In summary, although texture tracking in general has been investigated in the literature, the detection and tracking of texture patterns in real-time applications have not been fully exploited in existing algorithms.

For the purpose of texture pattern detection and object/material motion tracking, we propose discriminative and efficient texture representation methods by the combination use of binary descriptors and non-binary descriptors. Binary descriptors have two important ad-

vantages on local texture representation. One is their low computational cost, and the other is that the binary features of interest keypoints between video frames can be efficiently matched. Meanwhile, non-binary texture descriptors perform discriminative and stable representation ability. Combining these two types of descriptors implements a good trade-off between disciminative representation and efficient computation. Additionally, in order to implement accurate and efficient detection and tracking, we utilize both local texture appearances and global geometric structures of texture patterns underlying objects/materials. We will show the representation and tracking performance of our developed methods on the task of texture tracking.

2.2.2 Texture and Material Recognition

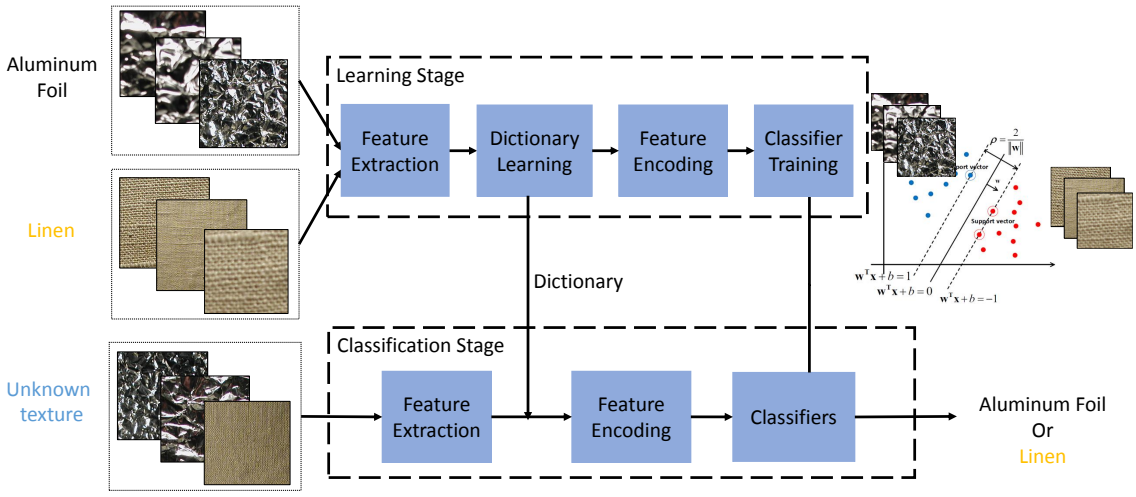


Figure 2.3: The BOW pipeline for texture classification.

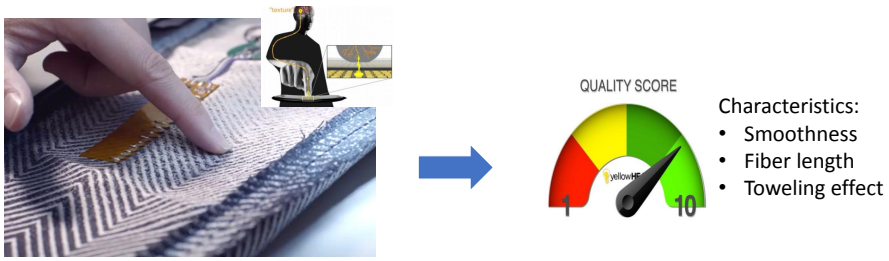
Texture classification relies on the reflectance properties of surfaces [28] or the statistical properties of pixels [58]. Texture classification has a variety of applications such as content-based image retrieval, object recognition, scene understanding, and bio-medical image analysis. Representing an image with a discriminative, compact feature vector is important to describe and distinguish materials "in the wild" with large intra-class visual variations and "at the fine-grained scale" with small inter-class appearance differences.

The main steps of texture classification are local feature extraction, feature encoding, and classifiers as shown in Fig. 2.3. Texture classification requires local texture descriptors to achieve two competing goals, discriminative description and low computational cost, especially for illumination, rotation, and scale variations. After local descriptors are converted into global image representations, the nearest neighbor classifier (NNC) or the support vector machine (SVM) is commonly used to classify images. Researchers also have proposed more complex classifiers, such as decision trees and random forests.

Though researchers have done lots of work on designing classifiers, we focus on extracting discriminative texture features and building powerful texture descriptors rather than developing classifiers and limit our study to use NNCs and SVMs. We summarize the literature of texture descriptors for texture classification in Table 2.2(d). To improve the recognition ability of texture descriptors, our work is developing texture representation algorithms in a progressive manner from handcrafted texture descriptors to generic descriptors via deep learning-based methods.

2.2.3 Material Surface Characterization

Material surface characterization is a challenging problem since some material characteristics of interest may be more latent than apparent, not always easily differentiated in the visual appearance. To handle this problem, human experts traditionally touch and feel material surfaces and provide scores for surface characteristics such as smoothness and fiber length shown in Fig. 2.4a. Instead of subjective evaluation with labor cost, automated material surface characterization facilitates generating characteristic ratings automatically. Provided a material sample, a flowchart of automatic material surface characterization is shown Fig. 2.4b, where a carefully designed imaging system and a computational feature extraction method are critical to the success of material surface characterization. Because of no public datasets for material surface characterization, we introduce a diverse dataset with various imaging condition changes and a controlled environment. In terms of an



(a) Subjective material surface characterization

(b) Automatic material surface characterization

Figure 2.4: Material surface characterization.

imaging system, since a regular digital camera is not able to capture fine detail on material surfaces, we apply microscopic imaging instead. Also, with the controlled environment, we simulate different settings including translation, rotation, and tilt changes. Though the combined use of an imaging system and a controlled environment, we create the first public dataset for material surface characterization with microscopic imaging.

Because of similar visual appearance between sample surfaces, we formulate automatic material surface characterization as a texture/material classification at fine-grained scale by treating different levels of texture characteristics as class labels. By using human rating as ground truth of class labels, supervised learning methods can be used here. For each surface characteristics such as smoothness, human ratings as 4 levels (i.e. class labels). In our work, we focus on designing computational feature extraction algorithms. We summarize the literature for feature extraction in Table 2.2(e). These methods range from traditional Tamura [38] features related to human perception to learning-based approaches [40, 15, 7]. However, Tamura features fail to differentiate various surface smooth levels. Learning-based material characterization work in the literature was reported by Sun et al. [7], who automatically estimated roughness of milled metal surfaces using a CNN.

However, his/her method does not generate a spatially invariant representation describing a distribution of texture features (i.e., feature encoding). FV-CNN [15] though involve a feature distribution for textures, the separation learning of feature extraction and encoding does not benefit from labeled data. Hence, aiming at improving the representation ability of texture representation approaches for material surface characteristics, we develop deep learning methods in an end-to-end manner with multi-level texture encoding, which simultaneously train feature extraction and encoding together and learn features from different layers and resolutions.

CHAPTER 3
BINARY REPRESENTATION IN TEXTURE TRACKING AND CLASSIFICATION

To describe local texture patterns presented in an image, binary representation, also referred as a local binary texture descriptor, is one of the widely used texture representation methods. Binary representation converts the intensity differences between pairs of neighboring pixels into binary values, which is discriminative, illumination invariant, and computationally efficient. As discussed in Sec. 2.1.2, these advantages of binary descriptors enable their appealing applications including tracking texture patterns with binary robust invariant scalable keypoints (BRISK) [54] and differentiating texture types with local binary patterns (LBP) [67]. In this chapter, we develop binary representation methods and study their role in texture tracking and texture classification tasks. First, we introduce the concept of local binary texture descriptors appropriate for texture tracking and classification, respectively. Secondly, we develop texture representation methods enabling a vision system to perform a series of visual tasks: (1) recognizing presented texture patterns, (2) tracking the motions of textures, and (3) recognizing various texture types. Finally, we summarize this chapter.

3.1 Local Feature Extraction with Binary Representation

3.1.1 Binary Representation for Texture Tracking

Binary representation such as BRISK [54] can track interest points (i.e., feature points) of images. For a local region of an image, a predefined sampling strategy as shown in Fig. 3.1 is applied to paired pixels around each center pixel of this region. According to the distance of each pixel pair, local gradient features are computed between long-distance pairs and are summed for determining a dominant orientation. Such operation of orienta-

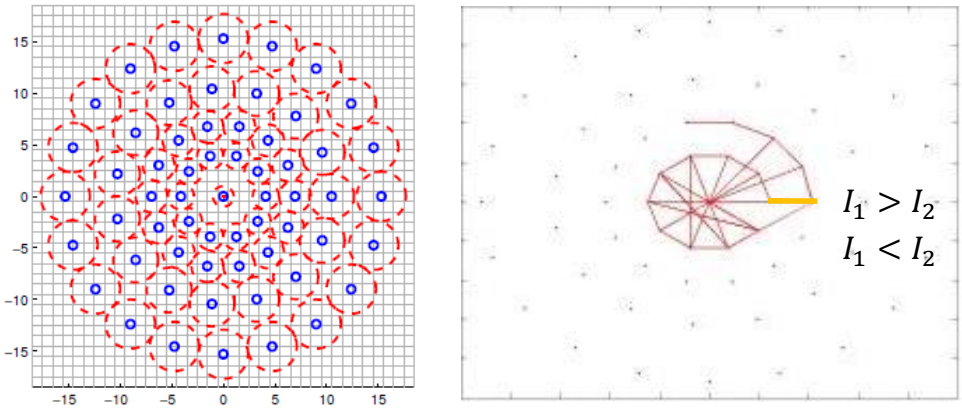
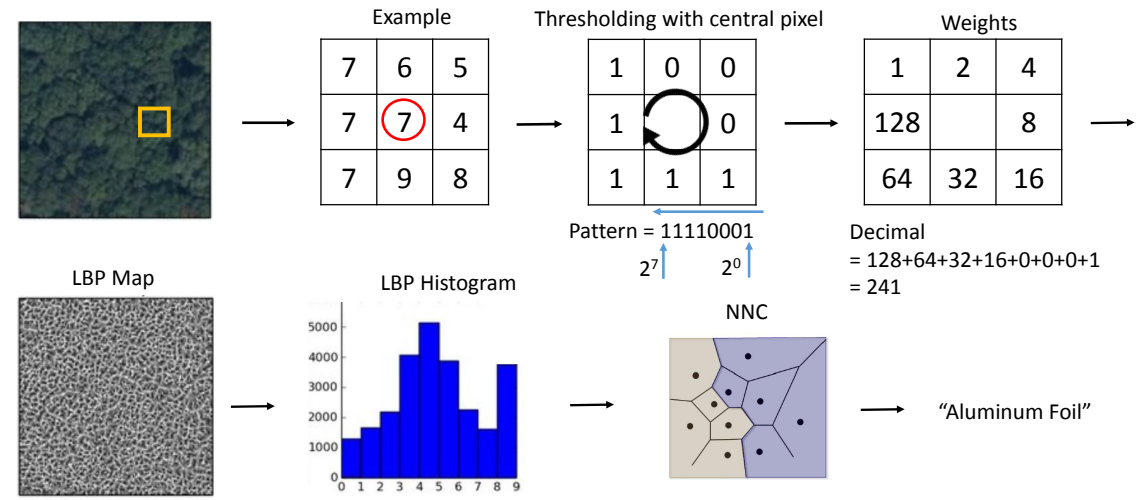


Figure 3.1: The concept of BRISK [54] binary representation.

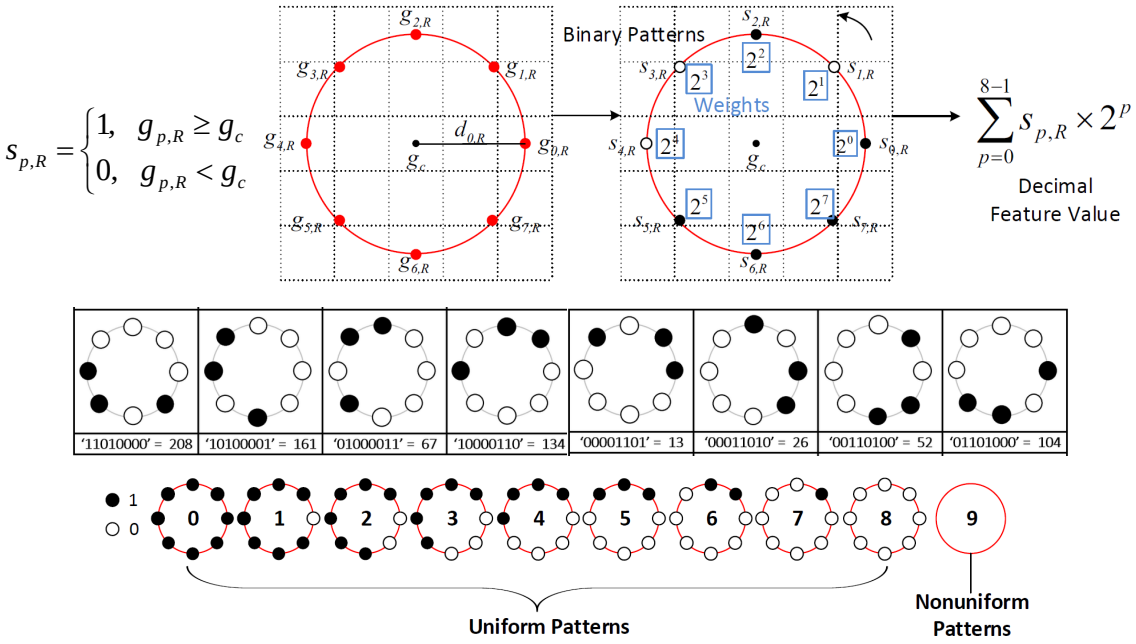
tion determination enables binary representation to recognize the same texture type with different rotation angles into one group. Meanwhile, the binarized intensity differences on short-distance pixel pairs are converted into binary patterns to extract textural information for each local region. After constructing binary descriptors for each feature point in an image, one set of feature points and their binary representations are obtained for this image. Motion tracking of textures requires the correspondence establishment between two sets of feature points for a pair of images. Because of the efficient bit operations of the Hamming distance between binary descriptors, binary representation has become a commonly used method for texture tracking.

3.1.2 Local Binary Patterns (LBP) for Texture Classification

We briefly discuss the concept of the original LBP and its application for texture classification. As shown in Fig. 3.2a, a binary pattern can be obtained for a 3×3 local region through the intensity comparison between a central pixel and its neighboring pixels. Multiplied by weights at corresponding pixels, the weighted values at each pixel are summed to a LBP code representing a pattern. Then, the distribution (i.e., histogram) of these LBP codes is used as an image presentation and is fed is fed into a standard classifier such as nearest neighbor classifiers (NNC) or support vector machines (SVM) for generating a class label. We provide an illustration of LBP formulation in Fig. 3.2b, where intensity values of



(a) LBP-based texture classification



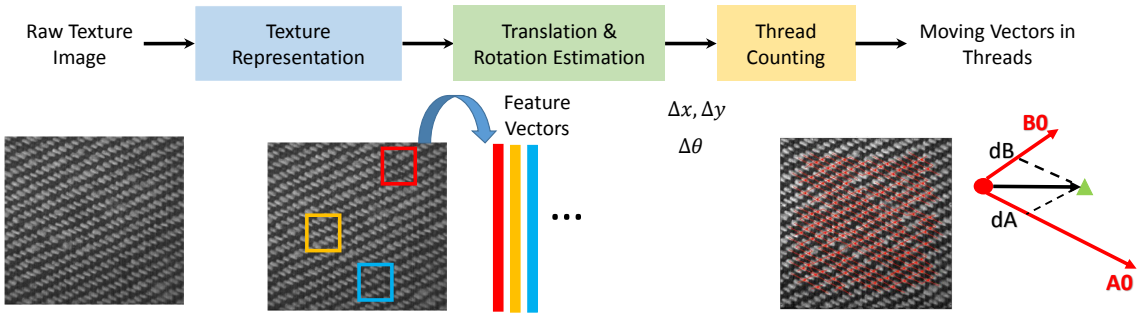
(b) LBP formulation

Figure 3.2: The basic concepts of LBP and LBP-based texture classification.

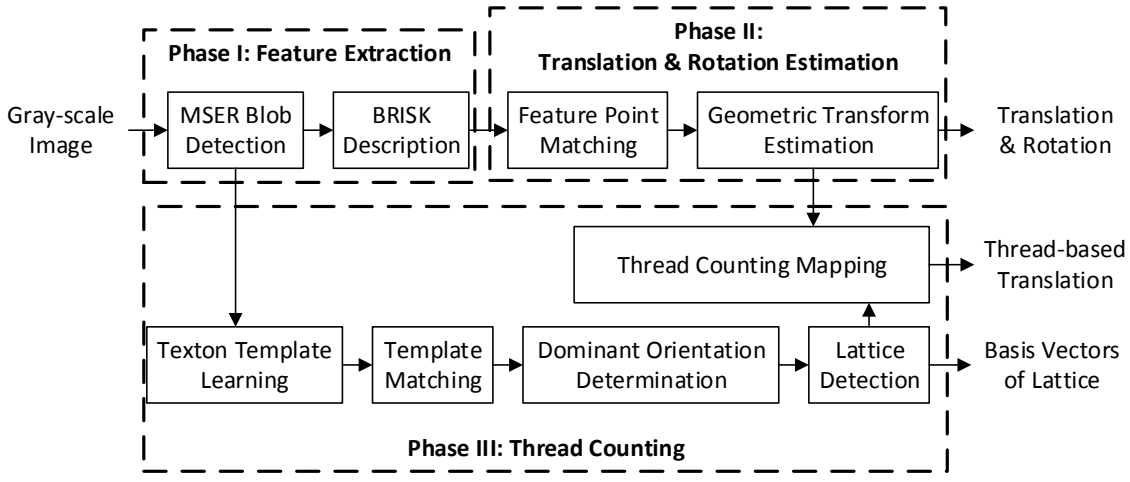
the center pixel and its neighbored are noted by g_c and $g_{p,R}$, the sign information of their intensity difference is denoted by $s_{p,R}$, p denotes location and R notes the radius of the sampling circle. To guarantee rotation invariance for texture classification, binary codes with the same circularly shifted format are grouped into one pattern, denoted $CLDP\_S^{ri}_{P,R}$, where "ri" represents rotation invariance. Taking an eight-bit binary pattern (i.e., $P = 8$) as an example, the total number of patterns is reduced from $2^8 = 256$ to 36 [26]. In addition,

researchers commonly apply uniform patterns ("$u2$"), which denote the frequency of bit-wise transitions from 0 to 1 or 1 to 0 in binary codes is less than two. Meanwhile, all other non-uniform patterns are grouped into one pattern. Such uniform and non-uniform grouping strategy characterizes local smoothness and achieves feature dimensionality reduction. "$riu2$" represents the pattern after the introduction of both the two grouping strategies: rotation invariance and uniform mapping [26]. When $P = 8$, the number of "$riu2$" patterns is reduced from $2^8 = 256$ to 10 with nine uniform patterns and one non-uniform pattern shown in Fig. 3.2b.

3.2 Binary Representation in Texture Tracking and Pattern Detection



(a) An illustration of texture tracking



(b) A flowchart of texture tracking

Figure 3.3: The texture tracking framework.

In this section, we introduce our developed binary representation methods and their role

in texture tracking and pattern detection. First, we introduces the concept and the objective of texture tracking as shown in Fig. 3.3a. Secondly, we develop a framework [39] shown in Fig. 3.3b to automatically track small motions and detect lattices for near-regular textures (e.g., denim textures in Fig. 3.3a).

3.2.1 Texture Tracking

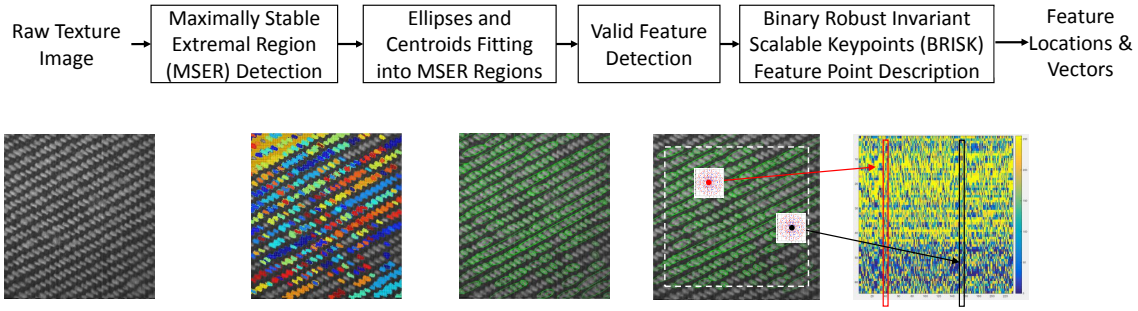
As shown in Fig. 3.3a, we aim at tracking the motions of texture patterns and reporting motions in both a camera-based coordinate system and a texture pattern-based coordinate system, ensuring more robustness to local texture deformation. Texture tracking from a video includes feature extraction, feature-point matching, and a geometric transformation with translation and rotation offsets.

Feature Extraction

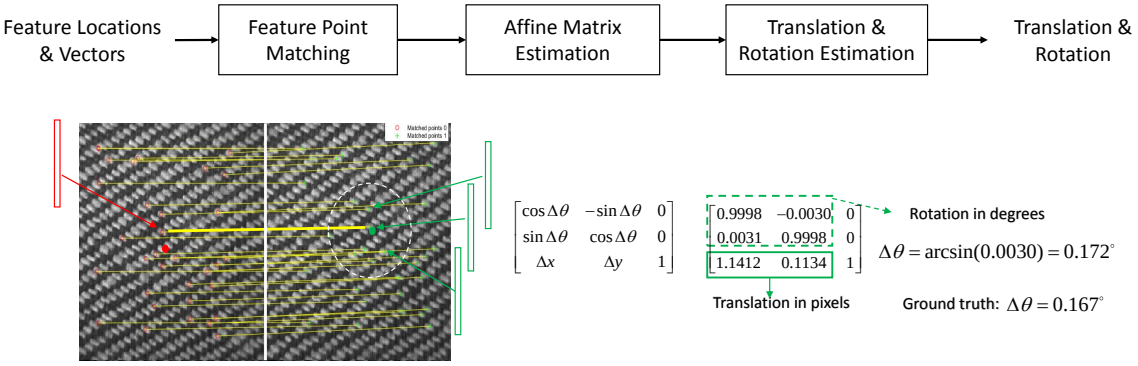
Since the image exhibits abundant bright blob regions with a near-regular placement and that a large percentage of the same blobs appear in successive frames of a video, we choose a maximally stable extremal region (MSER) [83] method for blob detection because of their stable region reorientation ability. Such stability is brought by the detection of covariant regions in MSERs. The detected MSERs are visualized in Fig. 3.4a, in which each unique color represents one individual MSER region. After MSER detection, we fit ellipses and centroids into detected blob regions displayed in green ellipses and central points in Fig. 3.4a. We utilize MSER regions later to match feature points between two consecutive video frames and to generate a texture template.

Each detection of each feature point in an image is followed by a step of feature description around this feature point to achieve the texture representation of each local region of this image. BRISK [54] as a binary texture representation offers a fast alternative to well-known algorithms such as scale-invariant feature transform (SIFT) and speeded up robust features (SURF) [53], and still maintains comparable representation abilities. We

extract a BRISK descriptor for each detected MSER feature point. In Fig. 3.4a, we use the red column vector to represent the BRISK feature vector of the red MSER feature point. It is worth noting that, for feature description, using typical texture features normally cannot satisfy both the robustness and the high-speed requirements simultaneously.



(a) Feature extraction



(b) Feature-point matching and geometric transformation estimation

Figure 3.4: Feature extraction, feature-point matching, and geometric transformation estimation.

Feature-point Matching and Geometric Transformation Estimation

Following feature-point detection and description, feature-point matching involves finding corresponding interest points between a pair of images. We use the Hamming distance to efficiently calculate the distance between binary patterns of two sets of feature points. We apply the nearest neighbor approach to the distances and set a match threshold for selecting the strongest matches. Fig. 3.4b illustrates the correspondence of feature points between a pair of images and a geometric transformation estimated from the locations of matched

pairs. In a word, with the discriminative representation ability of local feature descriptors, we build a reliable correspondence of the feature points between two video frames. Such reliable matching is a necessity to the tracking accuracy of texture motions.

In an affine matrix $\begin{bmatrix} cos \triangle \theta & -sin \triangle \theta & 0 \\ sin \triangle \theta & cos \triangle \theta & 0 \\ \triangle x & \triangle y & 1 \end{bmatrix}$ representing the geometric transformation between a pair of images, $\triangle x$ and $\triangle y$ note translation offsets in pixels on a camera-based coordinate system and $\triangle \theta$ (°) notes a rotation angle between the two images. To estimate the motion parameters in the affine matrix from a set of matched pairs that may contain outliers, we apply random sample consensus (RANSAC) to estimate a two-dimensional geometric transformation from matching pairs. Hence, we achieve a tracking of moving textures on a camera-based coordinate system with translation and rotation motion parameters.

3.2.2 Texture Pattern and Lattice Detection

Despite tracking motions in the camera space, the tracking of moving textures on the texture pattern-based coordinate system can bring more robustness to local texture deformation. Detection and tracking of texture patterns (i.e., thread counting) start with low-level visual cues (e.g., blob textures) in Fig. 3.4a and end with high-level lattice models shown in Fig. 3.5.

Texture Pattern Learning and Dominant Orientation Determination

We detect blob region candidates with MSERs in Sec. 3.2.1 shown in Fig. 3.4a, where some blobs are still connected, and others are obviously isolated. From attributes of each blob region such as its area and its intensity values, we group blobs into two clusters: individual blobs and grouping blobs. We utilize all individual blobs such as the highlighted blob regions to propose a blob template. By aligning all individual blobs with their centroids and averaging their intensity values, a blob template is learned. We use the template

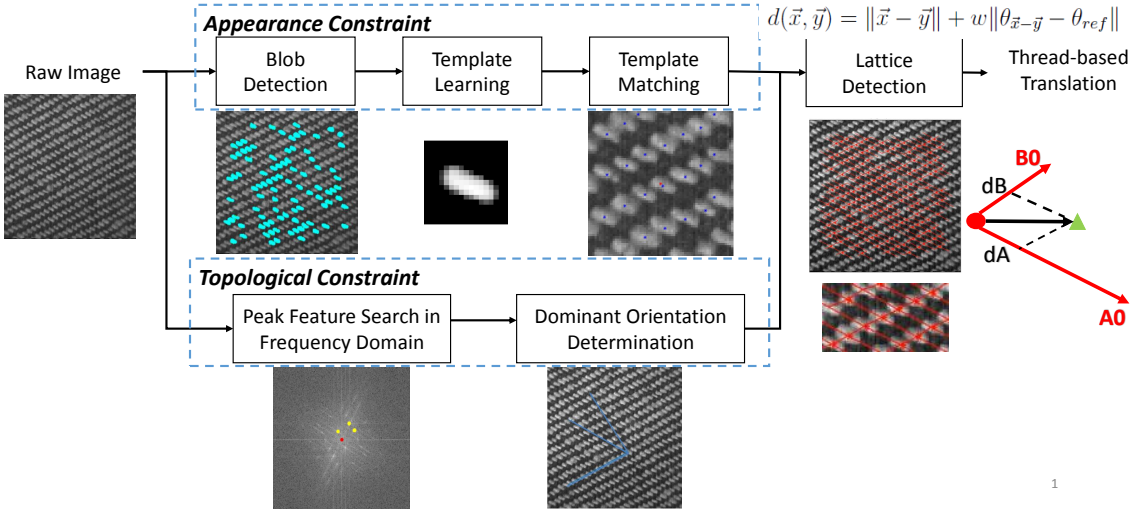


Figure 3.5: Texture pattern and lattice detection.

to detect the nearest neighboring blobs of the current MSER center(i.e., the red point in Fig. 3.5). To find neighboring blobs, we adopt the correlation-based template matching method, which utilizes the information in local peaks on a correlation map between the candidate neighboring blob region and the blob template. Fig. 3.5 illustrates the centroids of detected neighboring blobs in blue. Such blob template learning and matching is used as an appearance constraint for identifying texture patterns and determining the repetitive lattice.

Lattice Detection and Thread Counting

We apply both an appearance constraint and a topological constraint on the texture pattern and lattice detection. Proposing a lattice model represents determining a vector pair connecting the current blob centroid and its two nearest matched neighbors. Orientations of the two basis vectors should follow the guidance of obtained dominant directions, which results in global topological consistency. For potential vector pairs, we minimize the distance defined in Eq. 3.1 and generate a final lattice proposal.

$$d(\vec{x},\vec{y}) = \|\vec{x}-\vec{y}\| + w\|\theta_{\vec{x}-\vec{y}} - \theta_{ref}\|, \tag{3.1}$$

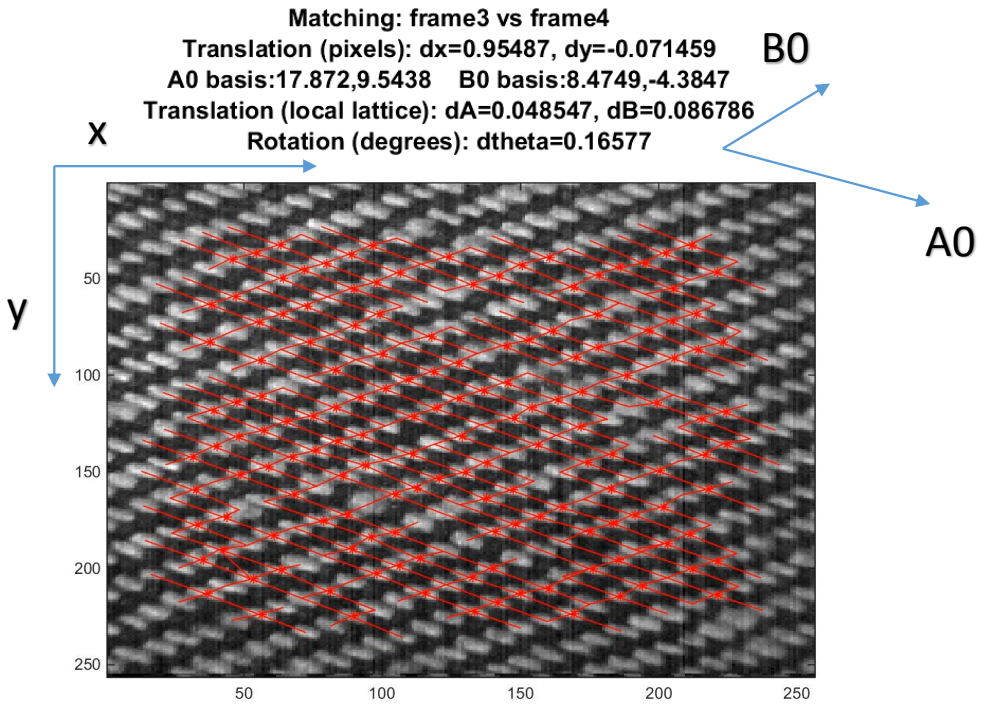


Figure 3.6: Texture tracking and thread counting illustration.

where $\vec{x}$ and $\vec{y}$ represent coordinates of the current blob centroid and the candidate blob centroid, respectively; $\theta_{\vec{x}-\vec{y}}$ denotes the angle of the vector connecting the current blob centroid and the candidate blob centroid; θ_{ref} represents a dominant direction of repetitive patterns estimated from the frequency domain; $\|\vec{x}-\vec{y}\|$ denotes a constraint of appearance similarities from template matching; and $\|\theta_{\vec{x}-\vec{y}}-\theta_{ref}\|$ notes a topological constraint. To balance the appearance constraint and the topological constraint, we use w, a weighting factor related to prior knowledge of weave patterns. By selecting candidate blobs with the smallest $d(\vec{x},\vec{y})$, we determine basis vectors along the two dominant directions. We superimpose the final proposal of a local lattice on inlier MSER feature points shown in Fig. 3.5. We produce the final lattice proposal by involving both the similarity of the pair-wise texture appearance and the global consistency of topological relationships.

To reduce the effect of fabric distortion and obtain fractional thread counting, we calculate the translation vector and decompose it into the local lattice coordinate system with the assumption that we have a prior knowledge of the fabric type and mapping information

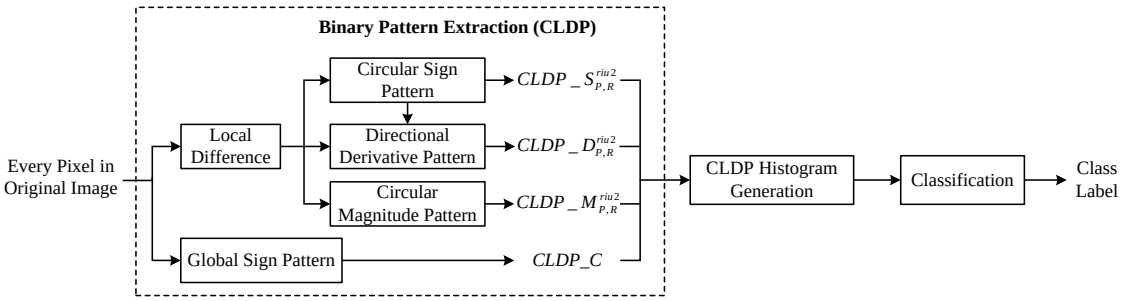


Figure 3.7: The block diagram of the CLDP-based texture classification.

between the local lattice and the physical fabric thread. Such a texture tracking method with a lattice detection module can handle local texture deformation and ensure accurate tracking of moving textures.

3.3 Binary Representation in Texture Classification

Binary representation is efficient not only for texture tracking, but also for texture classification. In this section, we develop discriminative binary representation and study their role in texture classification.

3.3.1 Completed Local Derivative Pattern (CLDP) Representation

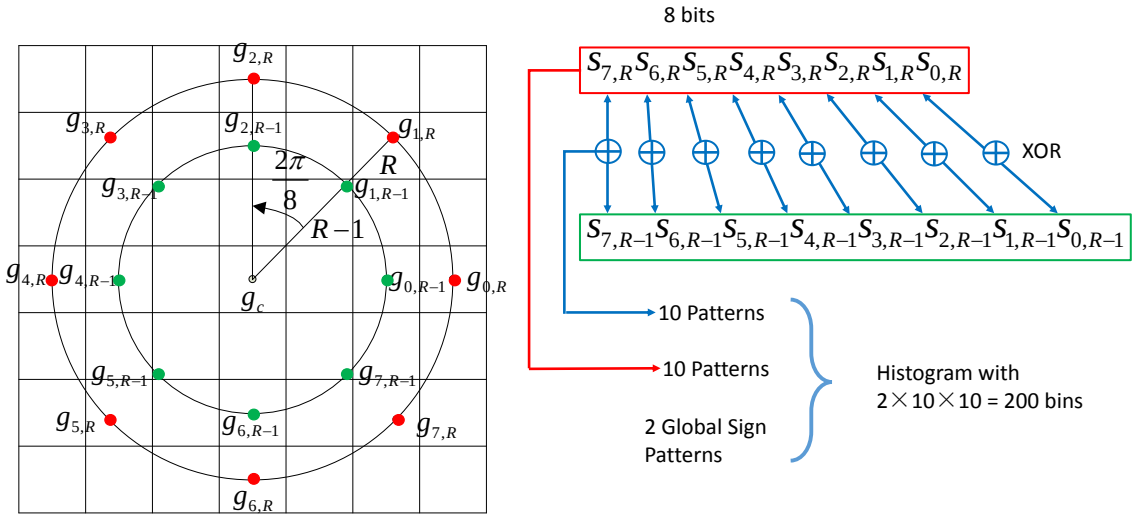


Figure 3.8: The sampling scheme for $P = 8$ with radii R and $R - 1$.

We develop a binary representation called completed local derivative pattern (CLDP) [60], to represent texture patterns in local regions by involving the derivative information. The block diagram of our CLDP-based texture classification is shown in Fig. 3.7. In contrast to completed local binary pattern (CLBP), which encodes local binary patterns in each scale separately, CLDP adds a new component, the directional derivative pattern, which involves the patterns of two neighboring scales in the same direction to calculate the corresponding directional derivative features. Such directional derivative pattern in CLDP characterizes local texture smoothness along each direction.

Feature Extraction of Local Texture Patterns

The intensity difference of neighborhood pixels has been widely used in texture representation due to its discriminative representation ability and its robustness to illumination changes. As Fig. 3.8 shows, each pixel g_c corresponds to P neighbors $g_{p,R}, p = 0, 1, \cdots, P-1$, which are evenly distributed on a circle with radius R. If $g_{p,R}$ does not have integer coordinates, its intensity value is estimated by bilinear interpolation. Local intensity difference is defined as $d_{p,R} = g_{p,R} - g_c$.

Circular Sign Pattern: According to $d_{p,R}$, we define the sign component of CLDP as $s_{p,R} = \begin{cases} 1, \ d_{p,R} \geq 0 \\ 0, \ d_{p,R} < 0 \end{cases}$ [69]. For simplification, we denote a thresholding function as $s_{p,R} = t\left(d_{p,R}, 0\right)$, where 0 represents the threshold. By applying function $t\left(d_{p,R}, 0\right)$ on radial directions, we obtain P binary bits and encode them in a counter-clockwise manner, denoted $CLDP\_S_{P,R}$.

As a grouping strategy of local binary patterns, a uniform pattern denoted $CLDP\_S^{riu2}_{P,R}$ is calculated:

$$CLDP\_S^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} s_{p,R}, \ U\left(CLDP\_S^{ri}_{P,R}\right) \leq 2 \\ P+1, \quad \text{Otherwise} \end{cases}, \tag{3.2}$$

where function $U(\cdot)$ counts the frequency of bitwise transitions.

Circular Magnitude Pattern: Since magnitude information $m_{p,R} = |d_{p,R}|$ of the local difference is complementary to sign information $s_{p,R}$, we obtain the magnitude component of CLDP by applying a thresholding function $t(m_{p,R}, c_{m,R})$ [69], where $c_{m,R}$ is the mean value of $m_{p,R}$ of all pixels in an entire image. Similar to the coding strategy in circular sign patterns, we define circular magnitude pattern $CLDP\_M^{riu2}_{P,R}$:

$$CLDP\_M^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} t(m_{p,R}, c_{m,R}), U\left(CLDP\_M^{ri}_{P,R}\right) \leq 2 \\ P+1, \qquad\qquad \text{Otherwise} \end{cases}. \tag{3.3}$$

Directional Derivative Pattern: To characterize texture smoothness along each direction, we introduce a new component, the directional derivative of local difference. Different from CLBP, which involves only one circle, we define two neighboring circles with radii R and $R-1$ as Fig. 3.8 shows. Based on sign components $s_{p,R}$ and $s_{p,R-1}$ from two neighboring circles, we define directional derivative pattern $CLDP\_D$:

$$CLDP\_D_{P,R} = \sum_{p=0}^{P-1} (s_{p,R} \oplus s_{p,R-1}) \cdot 2^p, \tag{3.4}$$

where operator $\oplus$ represents a bitwise exclusive OR (XOR) operation between the sign components of two neighboring circles along the same direction. In the outcome of $s_{p,R} \oplus s_{p,R-1}$, "1" means two local differences in one direction have different sign components, and it is likely that the intensity values of the two neighboring pixels vary remarkably. In contrast, "0" means the two local differences have the same sign component, which represents certain smoothness. To have the consistent coding format with $CLDP\_S^{riu2}_{P,R}$

and $CLDP\_M^{riu2}_{P,R}$, we define $CLDP\_D^{riu2}_{P,R}$ as follows:

$$CLDP\_D^{riu2}_{P,R} = \begin{cases} \sum_{p=0}^{P-1} \left(s_{p,R} \oplus s_{p,R-1}\right), U\left(CLDP\_D^{ri}_{P,R}\right) \leq 2 \\ P+1, \qquad\qquad \text{Otherwise} \end{cases}. \tag{3.5}$$

Because of frequently used "$riu2$" patterns, for simplification, we denote $CLDP\_S^{riu2}_{P,R}$, $CLDP\_M^{riu2}_{P,R}$, and $CLDP\_D^{riu2}_{P,R}$ as $CLDP\_S$, $CLDP\_M$, and $CLDP\_D$.

Global Sign Pattern: The intensity value of center pixel, g_c, reflects global information, but it is removed during the calculation of local intensity difference. To keep global information, we use a binary bit for each center pixel [69]: $CLDP\_C = t(g_c, c_I)$, where c_I denotes the mean intensity value of an entire image.

3.3.2 Scale-Selective Extended LBP (SSELBP) Representation

Since previous methods keep the settings of feature extraction unchanged, scale variations between images may degenerate texture representation. To increase the robustness to scale variations, we develop scale selective extended local binary patterns (SSELBP) [63] shown in Fig. 3.9.

Multi-scale Feature Extraction of Local Texture Patterns

Since we build SSELBP on top of ELBP, we give a brief introduction of ELBP [72]. To encode the intensity information, we define operator $ELBP\_CI$, which is the same as $CLDP\_S$ in Sec. 3.3.1. ELBP also involves operator $ELBP\_NI$ to extract information from the intensities of neighboring pixels, which is similar to $CLDP\_M$ in Sec. 3.3.1. The third operator involved in ELBP is $ELBP\_RD$, which encodes the intensity differences of pixels on two circles with radii R and $R'(R' < R)$ along radial directions similar to the spatial relationships shown in Fig. 3.8. Similar to $ELBP\_NI$, $ELBP\_RD$ generates a

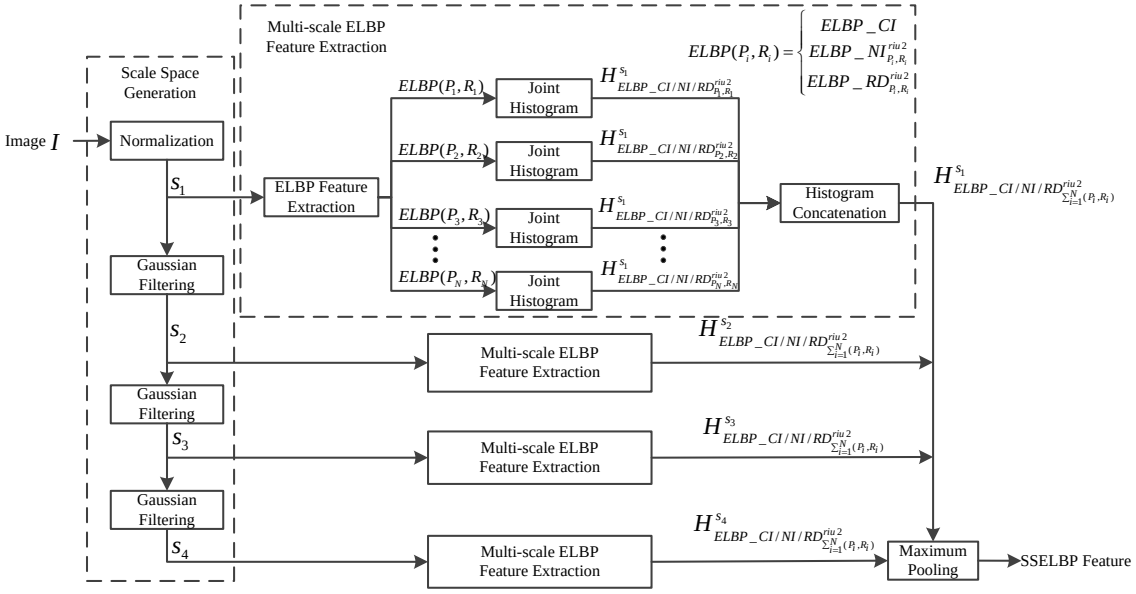


Figure 3.9: The block diagram of the SSELBP representation.

decimal value:

$$ELBP\_RD_{P,R}(x_c) = \sum_{p=0}^{P-1} s(g_{p,R} - g_{p,R'}) \cdot 2^p. \tag{3.6}$$

Provided the number of sampling points on each circle denoted P, operators $ELBP\_NI$ and $ELBP\_RD$ produce 2^P different binary patterns. To remove the rotation effect and reduce the pattern dimension, we apply rotation-invariant and uniform mappings and denote them as $ELBP\_NI^{riu2}_{P,R}$ and $ELBP\_RD^{riu2}_{P,R}$. For simplification, we denote ELBP containing patterns $ELBP\_CI$, $ELBP\_NI^{riu2}_{P,R}$, and $ELBP\_RD^{riu2}_{P,R}$ as $ELBP(P, R)$ in following sections.

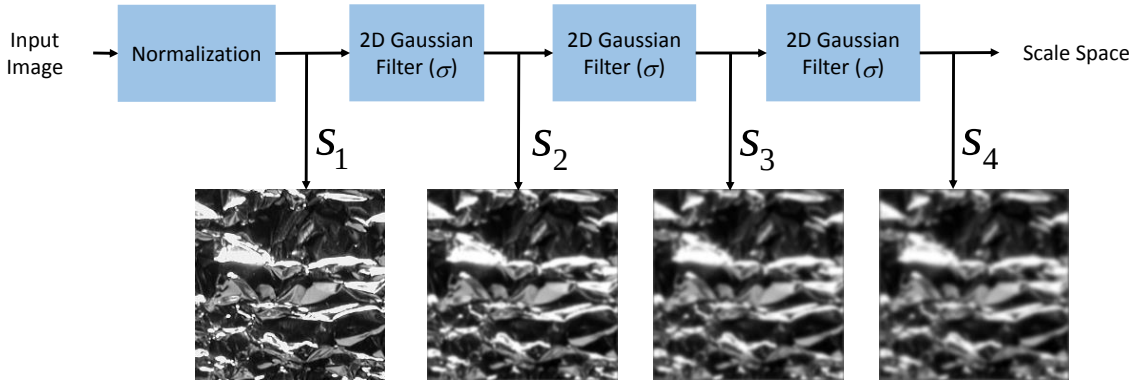
ELBP, which depends only on one or two local neighboring circles, is not robust to classify texture images with scale variations. To solve this problem, we use the multi-scale ELBP feature extraction method by involving various neighboring circles. Fig. 3.9 shows that with different (P, R) choices we obtain a group of ELBPs denoted $ELBP\,(P_i, R_i)$, $i = 1, 2, \cdots, N$, where N is determined based on the size and complexity of images. To combine patterns in $ELBP\,(P, R)$, we utilize the joint histogram that concatenates patterns and calculates the corresponding histogram. From another perspective, this combination

scheme is the conversion from a joint multi-dimensional histogram to a one-dimensional histogram. By defining this operation as "/", we denote the joint histogram of $ELBP\_CI$, $ELBP\_NI^{riu2}_{P_i,R_i}$, and $ELBP\_RD^{riu2}_{P_i,R_i}$ as $H_{ELBP\_CI/NI/RD^{riu2}_{P_i,R_i}}$. We concatenate the joint histograms of $ELBP(P_i, R_i)$, $i = 1, 2, \cdots, N$, and yield a multi-scale ELBP feature, $H_{ELBP\_CI/NI/RD^{riu2}_{\sum_{i=1}^{N}(P_i,R_i)}}$.

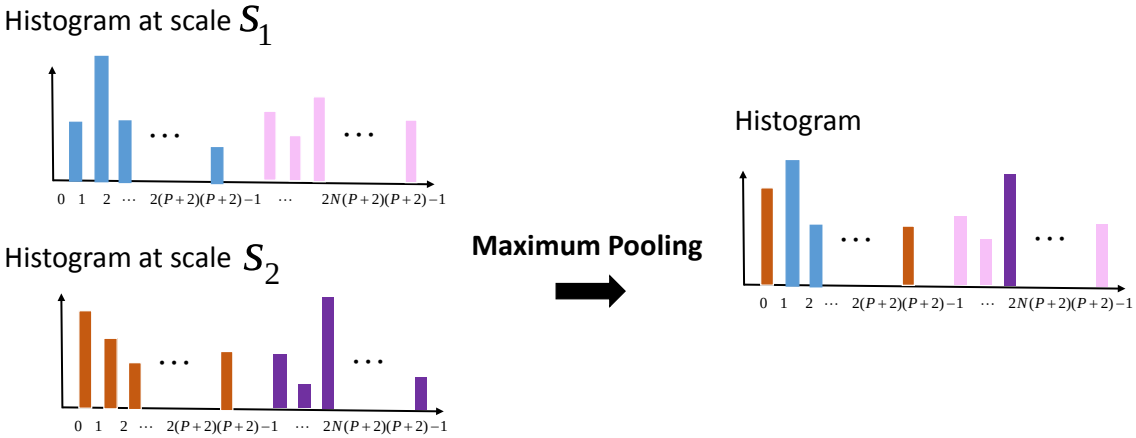
Scale Space-based Maximum Pooling

Scale Space Generation: To further increase the robustness of the developed method on images with scale variations, inspired by the SSLBP framework in [62], we build the scale space using Gaussian filters as shown in Fig. 3.10a to simulate images with scale variations. We normalize image I to ensure normalized image $\hat{I}$ has zero mean and unit variance. To build the scale space, we use Gaussian filters to smooth image $\hat{I}$ as follows: $s_l = \begin{cases} \hat{I}, & l = 1, \\ s_{l-1} * g(\sigma), & 1 < l \leq L, \end{cases}$, where $g(\sigma)$ defines a two-dimensional Gaussian filter with standard deviation σ and L represents the size of the scale space. s_l, $l = 1, 2, \cdots, L$, is the image at scale l. With the increase of l, more texture details are removed and the macro-structure of the texture becomes more significant. In Fig. 3.9, we set $L = 4$ empirically for illustration.

Maximum Pooling: To obtain scale-invariant texture features, we adapt the idea of the maximum pooling strategy as shown in Fig. 3.10b. For each scale s_l, we use the same (P, R) set to calculate the corresponding multi-scale ELBP histogram feature denoted $H^{s_l}_{ELBP\_CI/NI/RD^{riu2}_{\sum_{i=1}^{N}(P_i,R_i)}}$, $l = 1, 2, \cdots, L$. When combining these multi-scale features, we need to consider the robustness of the pooling result on texture images with scale variations. Because of the multiple choices of (P, R) in the multi-scale ELBP, we assume that the significant features of images at different scales can be captured by one parameter pair in the (P, R) set. When the scales of images change, these significant features still exist and can be captured by another parameter pair. Therefore, we utilize a maximum pooling



(a) Scale space



(b) Maximum pooling

Figure 3.10: Scale space-based maximum pooling.

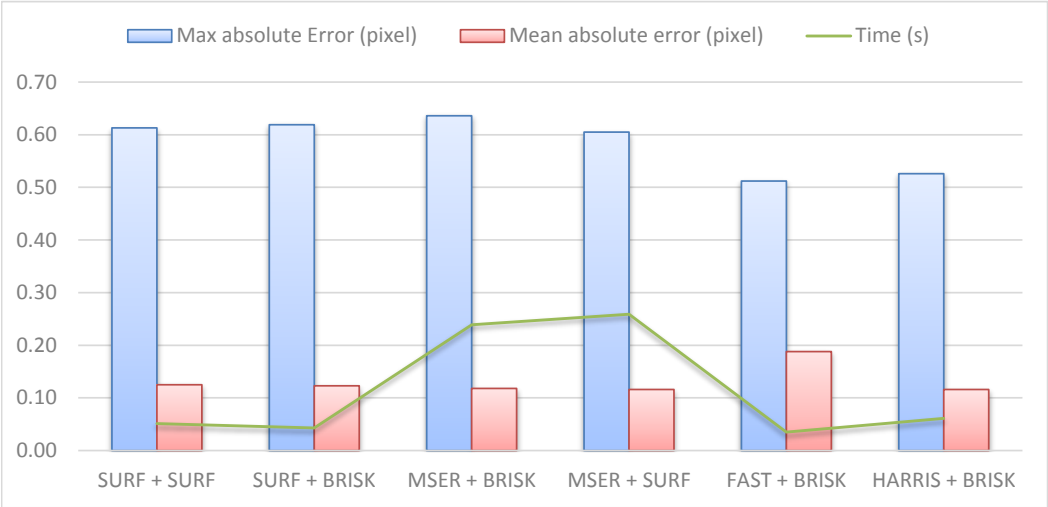
strategy that selects the maximum values from the corresponding bins of multi-scale ELBP histogram features at different scales as follows:

$$H_{ELBP\_CI/RD/NI_{\sum_{i=1}^{N}(P_i,R_i)}} = \max_{l=1,2,\cdots,L} \left(H^{s_l}_{ELBP\_CI/RD/NI^{riu2}_{\sum_{i=1}^{N}(P_i,R_i)}} \right). \tag{3.7}$$

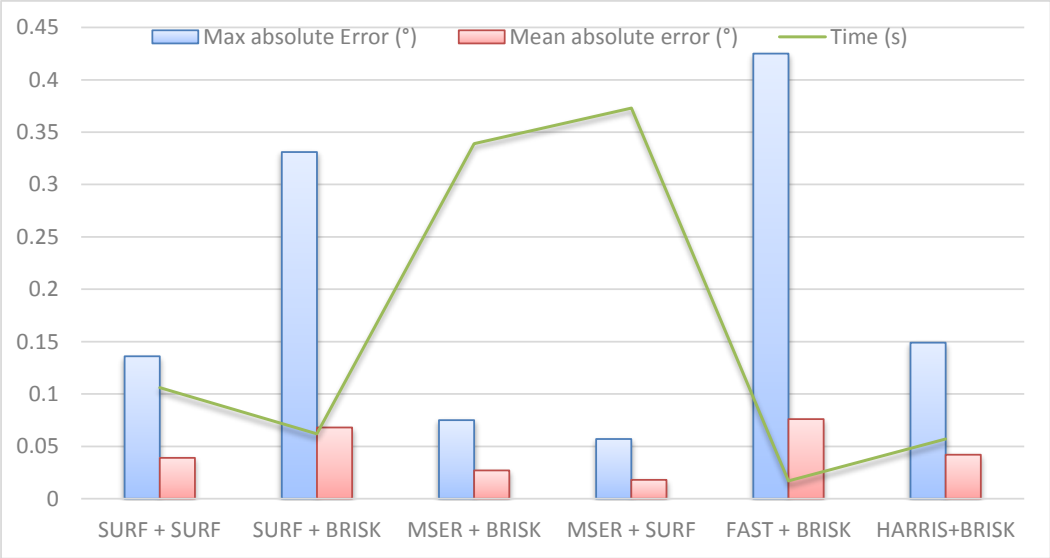
3.4 Experimental Results

3.4.1 Texture Tracking and Pattern Detection

To evaluate the tracking performance of our designed framework, we conduct a set of experiments, in which we estimated a translation offset in a camera space, a rotation angle, and a translation offset in a thread-based coordinate system between two frames with small



(a) Translation estimation



(b) Rotation estimation

Figure 3.11: Tracking accuracy and computational cost.

motions shown in Fig. 3.6. Our target texture is a piece of denim fabric mounted on a micro stage that allows precise translation and rotation. Since our camera captures only a small region of denim fabric, the field of view (FOV) of the captured images are filled with denim textures. The size of the images is 256×256. We implement algorithms on MATLAB®2014b with a PC (Intel i7-4790K, 4GHz, RAM: 32GB).

Translation Estimation: To evaluate the accuracy of translation estimation, we obtained

ground truth using the micro stage and conducted an extensive experiment. We translated the micro stage from zero to ten mm at intervals of 0.5 mm (i.e., around 7.53 pixels) in the horizontal direction and acquired 20 test images for our experiment. With the ground truth of translation offsets, we combined various feature detectors and descriptors (e.g. SURF [84], MSER [83], BRISK [54], FAST [85], and HARRIS [86]) and compared their estimation accuracy. To quantify accuracy for translation estimation, we use three metrics: (1) the maximum value of the absolute error between the estimated and actual translation values in pixels; (2) the mean value of the absolute error in pixels; and (3) the computational cost in seconds. We demonstrate the performance of the six algorithms on tracking translation in Fig. 3.11a, in which "A+B" denotes "feature detector+feature descriptor." From Fig. 3.11a, we observe: (1) "MSER+SURF" yields the lowest mean absolute error (i.e., 0.12 pixels); and (2) "FAST+BRISK" generates the lowest maximum absolute error (i.e., 0.51 pixels) and the lowest computational time (i.e., 0.035 seconds). To determine which algorithm to use, besides the three metrics mentioned above, we evaluate the accuracy and computation time of rotation estimation and thread counting.

Rotation Estimation: We applied a similar experimental setup to that in translation estimation into the evaluation of rotation estimation. By rotating the micro stage from 0° to 10° at intervals of $1/6^\circ$, we obtained 61 images and chose the first as a reference. The actual rotation angles between the test images and the reference image are successively $1/6^\circ, 1/3^\circ, \cdots, 10^\circ$. To evaluate the tracking of the rotation angles on tracking accuracy and computation time, we tested the same six feature extraction schemes as those tested in translation estimation. Since we simultaneously estimated translation and rotation parameters, the curves that exhibit computational cost in Figs. 3.11a and 3.11b present a similar shape and trend. Compared with other feature extraction methods shown in Fig. 3.11b, "MSER+BRISK" and "MSER+SURF" achieve superior tracking accuracy while sacrificing computational efficiency. Their mean absolute errors are 0.026° and 0.018°, respectively, and their maximum absolute errors are 0.075° and 0.057°, respectively. Among

Table 3.1: Comparison: our system versus the existing system of Book et al. [11].

	TmaxAE (pixel)	TmeanAE (pixel)	RmaxAE (°)	RmeanAE (°)	Time (s)	Thread
[11]	0.474	0.124	0.637	0.064	0.156	N/A
Ours [39]	0.636	0.118	0.075	0.026	0.339	Yes

all the comparison methods, "FAST+BRISK" consumes the least computation time (i.e., 0.017 seconds) but yields the highest error. In addition, by comparing feature detectors with the same feature extraction approach, we observe that MSER extracts higher quality and a larger number of blob features than SURF. Combined with the same feature detector, BRISK consumes less time than SURF. By involving tracking accuracy and computational efficiency, we choose "MSER+BRISK" for the thread-counting system.

Thread Counting and Comparison with the Existing System

The outcomes of thread counting include the basis vectors of a final lattice proposal in a camera space and two translation offsets in a lattice-based coordinate system. The mean error of translation estimation of a thread is about 0.02 (i.e., 1 thread = 0.33 mm). The vision system proposed by Book et al. [11] is the first and the only existing system for automatic garment sewing. Therefore, we compare the performance of their system and ours in Table 3.1, where TmaxAE, TmeanAE, RmaxAE, and RmeanAE represent translation maximum-, translation mean-, rotation maximum-, and rotation mean absolute errors, respectively. Our system outperforms theirs in two aspects: (1) our rotation tracking errors are significantly lower; and (2) our system performs thread counting, which is critical for a practical setting, but theirs cannot.

3.4.2 CLDP Representation on Texture Classification

To evaluate the performance of CLDP, we focus on the Outex [87] database and follow the testing protocol in [69]. The Outex dateset contains 24 classes of texture images with various rotations and illuminations. From the dataset, we select test suits TC10 and TC12,

in which the 24 classes of texture images are captured under 27 conditions, including three types of illumination conditions ("inca", "t184", and "horizon") and nine rotation angles ($0°$, $15°$, $30°$, $45°$, $60°$, $75°$, and $90°$).

Because of the efficiency and robustness of CLBP, we choose it as a benchmark and compare its classification accuracy with that of our developed method in different combination schemes as Table 3.2 shows. Each row in Table 3.2 represents the classification accuracy under a certain scheme with six types of (P, R) changing from $(8, 2)$ to $(24, 3)$. For each scheme of CLBP, we create a corresponding scheme that adds our newly introduced $CLDP\_D$ using joint or concatenated histograms. Each entry in Table 3.2 represents the average classification accuracy over three test suits mentioned above. We notice that CLDP improves the classification accuracy of CLBP counterparts for all (P, R) selections except for the last two schemes at the last two columns. Along with the changing of (P, R), CLDP with fewer sampling points has more accuracy improvement. This implicates that adding the directional derivative pattern is the most helpful when the number of sampling points is smaller. In addition, more complicated combination schemes correspond to less accuracy improvement. The superior performance of CLDP proves that $CLDP\_D$ describes the variations of texture along radial directions and provides complementary information to $CLDP\_S$, $CLDP\_M$, and $CLDP\_C$. Under scheme $CLDP\_S/M/D/C$ with $(P, R) = (8, 3)$, CLDP achieves its highest accuracy rate, 97.14%, which is 0.86% higher than that of CLBP, 96.28%, in scheme $CLBP\_S/M/C$ with $(P, R) = (24, 3)$. In addition, the feature dimension of $CLDP\_S/M/D/C$ with $(P, R) = (8, 3)$, $2 \times 10^3 = 2000$, is only one third of that of $CLBP\_S/M/C$ with $(P, R) = (24, 3)$, $26^2 \times 10 = 6760$. Therefore, CLDP with lower (P, R) has better performance in both accuracy and efficiency.

In addition to CLBP, we compare CLDP with other texture descriptors and list the classification accuracy on three test suits in Table 3.3. Since the training and testing data of TC12 have different illumination, we average the classification accuracy of the only two test suits in TC12 without involving that of TC10 for fair comparison. In Table 3.3, the upper

Table 3.2: Average classification accuracy (%) of CLBP and CLDP on TC10 and TC12.

Classification Accuracy (%)	(P, R)					
	$(8,2)$	$(8,3)$	$(16,2)$	$(16,3)$	$(24,2)$	$(24,3)$
$CLBP\_S$	77.67	80.94	82.40	85.89	84.07	87.04
$CLDP\_S/D$	86.01	90.69	89.44	92.03	90.18	92.55
Δ (Accuracy)	8.34	9.75	7.04	6.14	6.11	5.51
$CLBP\_M$	75.45	79.32	79.96	84.35	80.35	85.12
$CLDP\_M/D$	81.81	84.46	84.88	88.56	84.80	89.24
Δ (Accuracy)	6.36	5.14	4.92	4.21	4.45	4.12
$CLBP\_M/C$	88.03	88.37	91.79	92.36	91.47	93.15
$CLDP\_M/D/C$	91.48	91.42	92.83	94.39	92.37	94.57
Δ (Accuracy)	3.45	3.05	1.04	2.03	0.90	1.42
$CLBP\_S\_M/C$	92.11	92.31	93.41	94.52	93.51	94.94
$CLDP\_S\_D\_M/C$	92.40	93.87	93.68	95.24	93.69	95.35
Δ (Accuracy)	0.29	1.56	0.27	0.72	0.18	0.41
$CLBP\_S/M$	92.66	94.54	93.24	95.00	93.40	95.40
$CLDP\_S/M/D$	94.87	96.43	95.07	96.26	93.67	95.59
Δ (Accuracy)	2.21	1.89	1.83	1.26	0.27	0.10
$CLBP\_S/M/C$	95.41	96.08	95.44	96.16	95.19	**96.28**
$CLDP\_S/M/D/C$	96.29	**97.14**	96.25	96.45	94.94	95.97
Δ (Accuracy)	0.88	1.06	0.81	0.29	-0.25	-0.31

and lower parts correspond to uni- and multi-scale operators, respectively. We notice that, the develped method, being a uni-scale operator, has the best performance when compared with other uni-scale state of the art. In addition, the developed CLDP can also achieve comparable accuracy with multi-scale operators, which increase accuracy by sacrificing computational efficiency.

3.4.3 SSELBP Representation on Texture Classification

To build a scale space, we use Gaussian filters with scale parameter $\sigma = 2^{0.25}$ and empirically set the size of the scale space to be four. For all scales, we extract multi-scale ELBP features using the same set (P_i, R_i), $i = 1, 2, \cdots, N$. To reduce the feature dimension, we set $P = 8$ for all radii and select N radii from set $\{1, 2, \cdots, 8\}$. The selection of R' in the calculation of $ELBP\_RD$ also depends on R. If we use four radii (R_1, R_2, R_3, R_4) to calculate the multi-scale ELBP features, the corresponding R' should be (R_0, R_1, R_2, R_3),

Table 3.3: Classification accuracy (%) of other texture descriptors on TC10 and TC12. Accuracies for PRICoLBP$_g$ which are taken from the work by Liu et al. [61].

Classification Accuracy (%)	TC10	TC12		
		t184	horizon	Average
CLBP [69]	99.30	95.32	94.54	94.93
DLBP + NGF [70]	99.10	93.20	90.40	91.80
PRICoLBP$_g$ [88]	94.48	92.57	92.50	92.54
LDDP [73]	97.89	95.30	93.40	94.35
CLBC [75]	98.80	94.00	94.07	94.04
DNS + LBP$_{(24,3)}$ [89]	99.27	94.40	93.85	93.63
PLBP_C [90]	98.95	95.32	94.95	95.14
NTLBP [91]	99.24	96.18	94.28	95.23
CLBP_S/M/C$_{(8,1)+(16,2)+(24,3)}$ (Multi-scale) [69]	99.14	95.18	95.55	95.37
LDDP_1/2$_{(8,1)+(16,2)+(24,3)}$ (Multi-scale) [73]	98.64	95.9	94.16	95.08
CLBC_S/M/C$_{(8,1)+(16,2)+(24,3)}$ (Multi-scale) [75]	99.38	94.98	95.51	95.25
MSJ-LBP (Multi-scale) [77]	96.53	94.95	96.34	95.65
pi-LBP (Multi-scale) [76]	99.17	95.72	94.54	95.13
pi-LBP/C (Multi-scale) [76]	98.96	97.36	97.11	97.24
CLDP (Our method) [60]	99.32	96.55	95.63	96.09

where R_0 refers to the central pixel. To investigate the influence of N on classification accuracy, we test SSELBP on the KTH-TIPS database and list results in Table 3.4. The best classification accuracy of our method on the KTH-TIPS database is 98.11% with radius selection $(2, 3, 4, 7)$. When N equals four or five, we obtain higher accuracy with smaller deviations. The increase of N sacrifices the feature dimension but does not improve classification accuracy.

We compare the classification accuracy of SSELBP with those of state-of-the-art texture descriptors. Table 3.5 lists classification results and indicates the classifier each method uses. The number in the bracket following databases denotes the number of training samples used per class. Because of the efficiency and robustness of SSLBP, we choose it as a benchmark and compare its classification accuracy with that of SSELBP. For the KTH-TIPS database, SSELBP achieves the best accuracy 98.11% among all sampling schemes,

which is 0.31% higher than SSLBP. For the UMD database, the classification accuracy of SSELBP is 98.96%, which is 0.12% higher than that of SSLBP. In addition to SSLBP, we compare SSELBP with other texture descriptors such as CLBP, random projection (RP), and median robust ELBP (MRELBP). The performance of our developed SSELBP with a feature dimensional of 800 is comparable to other texture descriptors while the feature dimensions of CLBP and SSLBP are 2200 and 2400, respectively.

Table 3.4: Classification accuracy ($\%$) of SSELBP using different sampling schemes on the KTH-TIPS database.

Radius Number (N)	Maximum Accuracy ($\%$)	Radius Selection for Maximum	Mean Accuracy ($\%$)	Standard Derivation	Feature Dimension
1	96.44	(2)	94.80	1.56	200
2	97.86	$(1, 6)$	97.04	0.63	400
3	98.09	$(2, 5, 8)$	97.51	0.43	600
4	**98.11**	$(2, 3, 4, 7)$	97.71	0.30	800
5	98.10	$(1, 2, 3, 4, 8)$	97.84	0.20	1000

Table 3.5: Classification accuracy ($\%$) of SSELBP and other texture descriptors on KTH-TIPS and UMD databases.

Classification Accuracy ($\%$)	KTH-TIPS (40)	UMD (20)
CLBP (NNC) [69]	97.19	98.00
RP (NNC) [92]	97.71	99.13
MRELBP (NNC) [61]	-	98.66
SSLBP (NNC) [62]	97.80	98.84
SSELBP (NNC) (Our method)	98.11	98.96

3.5 Summary

In this chapter, we investigated the role of binary representation in texture tracking and classification. The performance of texture tracking depends heavily on the binary representation of interest points, which are commonly identified by a detector with specific criterion. The binary representation of interest points is required to have sufficient discriminative power, which determines the matching accuracy of corresponding interest points in two adjacent frames. In addition to texture tracking, the discriminative power of binary

representation is also crucial for texture classification. Binary representation algorithms (e.g., LBP variants) typically extract texture patterns in binary from densely sampled image patches over an image and utilize the corresponding distribution to generate full image representation. Our contributions on binary representation and its applications in texture tracking and classification are summarized as follows:

First, to achieve motion tracking of textures, we developed an efficient, robust, and accurate feature-based approach to track texture patterns and provided associated motion information in terms of position and orientation. Our contributions include the following aspects: (1) an innovative framework that integrates a lattice detection module to accomplish texture tracking in a texture pattern-based coordinate system instead of a pixel-based system, to ensure robustness to local texture deformation; (2) a novel algorithm for fast and efficient lattice detection, achieved by constraining on local appearance similarities and global topology; and (3) an extensive comparative study evaluating various methods of interest point detection and description for their applicability to the texture tracking problem, demonstrating its high potential for real-time visual tracking problems. During the process of texture tracking, given an appropriate interest point detector, the selection of interest point descriptor is an important step for accurate interest point matching and highly computational efficiency. Binary representation (e.g. BRISK) used in this chapter is a good option because of its fast feature extraction and distance calculation. Therefore, for the problem of real-time texture tracking, binary representation performs well on describing local textures in terms of disriminative ability and computational efficiency.

Secondly, to better involve multi-scale information or handle scale variations in the task of texture classification, we developed two binary representation approaches. First, different from popular LBP variants like CLBP encoding local binary patterns in each scale separately, we developed a texture descriptor called CLDP to involve multi-scale textural information in local regions. CLDP introduces a new component, the directional derivative pattern to extract the patterns of two neighboring scales in the same direction to calcu-

late the corresponding directional cross-scale correlation, which characterizes local texture smoothness along each direction. By including the new component, CLDP covers four types of binary representations: the sign, magnitude, and local directional derivative of local intensity differences and the intensity values of center pixels, respectively. For a local region around one pixel, the information from these four binary representations is fused though a joint encoding of binary codes and such encoding encodes the combined information into a feature value representing the pattern of this local region. When compared with state-of-the-art uni- and multi-scale texture descriptors on the Outex database [87], CLDP outperforms its uni-scale counterparts like CLBP in terms of texture classification accuracy without adding computational complexity and it is comparable to the multi-scale texture descriptors on the classification accuracy, which supports our claim that binary representation involving multi-scale information improves its discriminative ability of textural information.

Thirdly, since commonly used binary representation like ELBP keeps the settings of feature extraction unchanged for all images, scale variations between images may degenerate the classification performance of binary representation. Therefore, we developed a binary representation approach named scale-selective extended local binary patterns (SSELBP) to generate an image representation robust to scale variations. SSELBP builds a scale space through applying Gaussian filters and extracts binary patterns at each scale. Similar to CLDP, the representation for each scale utilize binarized local intensity differences and involves multi-scale information. For the purpose of scale invariance, a key step of SSELBP is a maximum pooling strategy on the features across all scales. This step obtains scale-invariant features. With the experimental results on two texture databases with scale variations, we observed that compared to state-of-the-art descriptors, SSELBP achieved comparable accuracy with lower-dimensional features. Therefore, binary representation plays an important role in discriminative, efficient, and robust representation for texture classification.

Table 3.6: Denotations for Chapter 3.

Denotations	Meaning	First Appeared
g_c	the intensity value of the center pixel	Sec. 3.1.2
$g_{p,R}$	the intensity values of neighboring pixels	Sec. 3.1.2
$s_{p,R}$	the sign information of their intensity difference	Sec. 3.1.2
$CLDP\_S^{ri}_{P,R}$	binary codes with the same circularly shifted format are grouped into one pattern	Sec. 3.1.2
ri	rotation invariance	Sec. 3.1.2
$u2$	uniform patterns	Sec. 3.1.2
$riu2$	the pattern involving rotation invariance and uniform mapping	Sec. 3.1.2
$\vec{x}$ and $\vec{y}$	coordinates of the current blob centroid and the candidate blob centroid, respectively	Sec. 3.2.2
$\theta_{\vec{x}-\vec{y}}$	the angle of the vector connecting the current blob centroid and the candidate blob centroid	Sec. 3.2.2
θ_{ref}	a dominant direction of repetitive patterns estimated from the frequency domain	Sec. 3.2.2
$\|\vec{x}-\vec{y}\|$	a constraint of appearance similarities from template matching	Sec. 3.2.2
$\|\theta_{\vec{x}-\vec{y}}-\theta_{ref}\|$	a topological constraint	Sec. 3.2.2
$d_{p,R}$	local intensity difference	Sec. 3.3.1
$CLDP\_S^{riu2}_{P,R}$	circular sign pattern: binary codes with $riu2$ pattern grouping strategy	Sec. 3.3.1
$CLDP\_M^{riu2}_{P,R}$	circular magnitude pattern: binary codes with $riu2$ pattern grouping strategy	Sec. 3.3.1
$m_{p,R}$	the magnitude information of the local difference	Sec. 3.3.1
$c_{m,R}$	the mean value of $m_{p,R}$ of all pixels in an entire image	Sec. 3.3.1
$CLDP\_D^{riu2}_{P,R}$	directional derivative pattern	Sec. 3.3.1
$CLDP\_C$	global sign pattern	Sec. 3.3.1
$ELBP\_CI$	the same as $CLDP\_S$	Sec. 3.3.2
$ELBP\_NI$	extract information from the intensities of neighboring pixels; similar to $CLDP\_M$	Sec. 3.3.2
$ELBP\_RD$	encodes the intensity differences of pixels on two circles along radial directions	Sec. 3.3.2
$ELBP\,(P_i, R_i)$	a group of ELBPs	Sec. 3.3.2
$H_{ELBP\_CI/NI/RD^{riu2}_{P_i,R_i}}$	the joint histogram of $ELBP\_CI$, $ELBP\_NI^{riu2}_{P_i,R_i}$, and $ELBP\_RD^{riu2}_{P_i,R_i}$	Sec. 3.3.2
$H^{s_l}_{ELBP\_CI/NI/RD^{riu2}_{\sum_{i=1}^{N}(P_i,R_i)}}$	the multi-scale ELBP histogram feature at scale s_l	Sec. 3.3.2

CHAPTER 4
DISCRIMINATIVE MULTI-SCALE TEXTURE REPRESENTATION IN TEXTURE CLASSIFICATION

According to our discussion in Chapter 3, an effective binary representation method improves the accuracy and efficiency of texture classification. However, the binarization of local intensity difference features may lead to information loss of local textures. To reduce this problem, we present an efficient and distinctive local descriptor, namely block intensity and gradient difference (BIGD) in this chapter. In an image patch, we randomly sample multi-scale block pairs and utilize the intensity and gradient differences of pairwise blocks to construct the local BIGD descriptor. The random sampling strategy and the multi-scale framework help BIGD descriptors capture the distinctive patterns of patches at different orientations and spatial granularity levels. We use vectors of locally aggregated descriptors (VLAD) or improved Fisher vector (IFV) to encode local BIGD descriptors into a full image descriptor, which is then fed into a linear support vector machine (SVM) classifier for texture classification. We compare our developed descriptor with typical and state-of-the-art ones by evaluating their classification performance on public texture datasets.

4.1 Multi-scale Local Feature Extraction and Image Representation

4.1.1 Local Difference Feature Extraction

Binarized Local Difference Feature Extraction

The discriminative ability and computational efficiency of a texture descriptor are critical to many computer vision applications. Assuming that small patches usually exhibit texture patterns or structures, a discrimnative information for a patch can be obtained by a texture descriptor. To achieve this purpose, a highly efficient and discriminative binary descriptor

named local difference binary [68] computes a binary string for an image patch by testing simple intensity and gradient differences test on pairwise grid cells within the patch. Based on prior related work, such intensity difference capture appearance variations for a patch. And a multiple-gridding strategy is also applied to capture texture patterns within the patch at different spatial granularities. These factors facilitate achieving high computational speed and robustness as state-of-the-art binary descriptors. Here we briefly discuss the two key factors: binary tests and a sampling strategy.

First, to keep the discriminative ability of describing an image patch, both average intensity $\bar{I}$ and first-order gradients d_x and d_y of grid cells within the patch are computed. To capture patterns of the patch, a set of binary tests compares $\bar{I}$, d_x, and d_y of a pair of grid cells, thus providing a more discriminative representation than other binary descriptors comparing only $\bar{I}$. Additionally, such regional information of small blocks are more robust to noise than raw pixel intensity values. Given that a thresholding function is defined as $t(a,b) = \begin{cases} 1, \ a > b, \\ 0, otherwise, \end{cases}$, a binary test is defined as:

$$\tau(a,b) = t(f(a), f(b)) = \begin{cases} 1, \ f(a) > f(b), \\ 0, otherwise. \end{cases} \tag{4.1}$$

where f is a function for extracting $\bar{I}$, d_x, or d_y from a grid cell.

Second, a multiple-gridding strategy encodes patterns within the patch at different spatial granularities. Coarse-level grids cancel out high frequency noise while fine-level grids capture detailed information, thus enhancing its discriminative ability. We illustrate an example of such feature extraction for an image patch in Fig. 4.1. The image patch includes 3×3 grids. We show binary tests for three pairs of grid, respectively. For each pair of grids, the intensity value (I), gradient in x and y directions (d_x and d_y) are calculated for each grid and then compared to generate binary strings. The three examples of binary tests show that binarized local differences well differentiate three patterns even under the case when

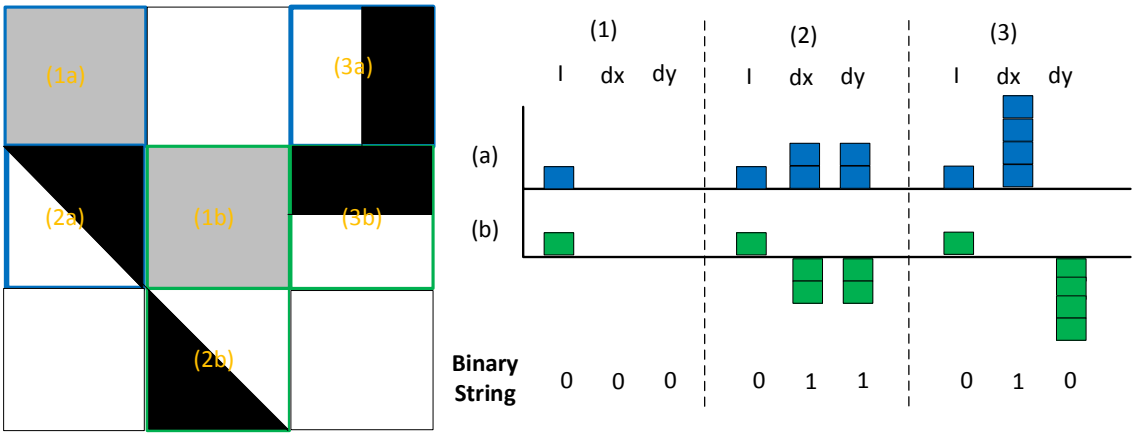


Figure 4.1: The feature extraction of binarized local differences with multiple gridding for an image patch.

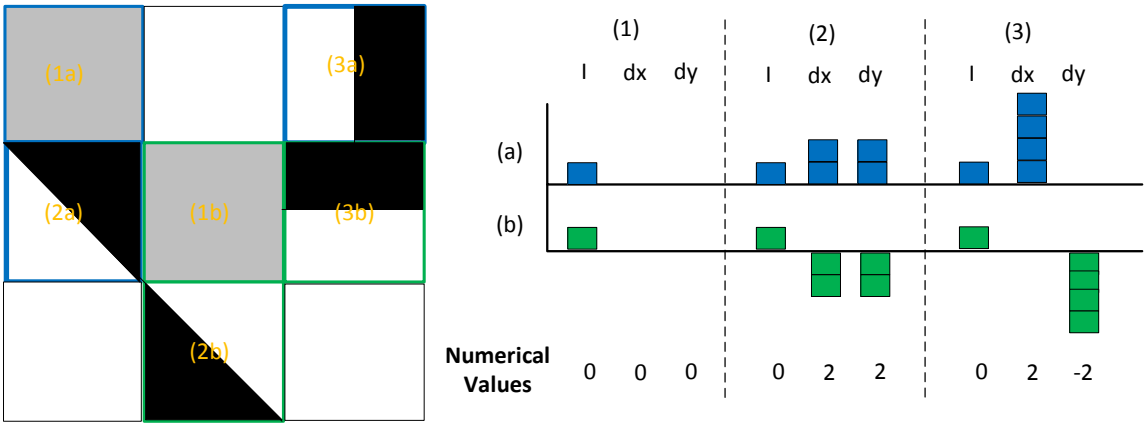


Figure 4.2: The feature extraction of non-binarized numerical local differences with multiple gridding for an image patch.

a pair of grids have the same intensity value but different directional gradients. Though Fig. 4.1 illustrates a gridding strategy of 3×3 grids, various layout of grids can be applied to capture textual information under different spatial granularities.

However, the binarization of local intensity differences leads to the loss of intensity information, which weakens the ability of discrimination. Another disadvantage of binary descriptors is their dimensions, which will grow exponentially when the number of pairwise comparisons on neighboring pixels increases.

Non-binarized Difference Feature Extraction

To encode the local structure of a patch, dense micro-block difference (DMD) [64] was proposed, which composes of real-valued intensity differences instead of binary codes of different micro-blocks in local patches. As illustrated in Fig. 4.2 for a single resolution scale s, given a patch P of size $p \times p$ and two sets of sampling points, $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and $Y = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N\}$, the feature vector $v_s(P)$ of numerical intensity differences for the pair of microblocks of size s is defined as:

$$v_s(P) = \{f_s(\mathbf{x}_1) - f_s(\mathbf{y}_1), f_s(\mathbf{x}_2) - f_s(\mathbf{y}_2), \cdots, f_s(\mathbf{x}_N) - f_s(\mathbf{y}_N)\}, \tag{4.2}$$

where $f_s(\mathbf{x}_1)$ is the average value of pixel intensities in the microblock located at the position $\mathbf{x}_1$. These features are normalized within the patch to avoid illumination variations. The formulation for a single scale is easily extended to multiple scales by a concatenating operation of feature vectors and is easily applied to multiple orientations by a random sampling strategy. Although such dense microblock difference captures non-binarized patch-based features at multiple granularities and orientations, the neglect of first-order gradients may deteriorate the discriminative ability. Therefore, we consider to generalize the function f to include more discriminative features such as gradient-based features dx and dy.

4.1.2 Image Representation for Numerical Local Feature Extraction

Standard Pipeline for Texture Representation

As reviewed in [14, 82, 45], texture representation techniques include filter-bank-based approaches, statistical models, bag of words (BOW) pipelines, and the latest CNN-based descriptors. Over the last decade, representation based on the BOW model has become a popular choice over others. BOW combined with local descriptors, such as the scale-invariant feature transform (SIFT) [27], local binary pattern (LBP) [44], or LBP variants [60, 93], was the most widely used texture representation method. By assigning each local descriptor

to its nearest visual word (i.e., a hard assignment), the BOW encoder calculates a histogram of visual word occurrences. To include richer information instead of simple occurrences, two popular extensions of BOW are vector of locally-aggregated descriptors (VLAD) [55] and Fisher vectors (FV) [34]. Different from BOW, VLAD accumulates the differences between a visual word and its corresponding local descriptors to aggregate first-order statistics of descriptors, while FV encodes both first- and second-order statistics of descriptors. However, these techniques still require further development because of various challenging real-world scenarios.

VLAD Encoding

To encode the BIGD descriptors of all patches into a full image descriptor, we utilize a typical encoding method, VLAD [55], which is the simplified form of the Fisher vector (FV). Following the conventional notations of VLAD encoding, we denote BIGD descriptor $\mathbf{v}^{\text{BIGD}}_{(X,Y),S}$ as $x \in \mathbb{R}^d$, where d represents the dimension of $\mathbf{v}^{\text{BIGD}}_{(X,Y),S}$. We first partition BIGD descriptors extracted from the patches of training images into K clusters using k-means clustering. The corresponding cluster centers, denoted $\{u_i\}_{i=1}^{K}$, $u_i \in \mathbb{R}^d$, as codewords, construct a codebook. Then from each image, we assume that we extract m BIGD descriptors, denoted $\chi = \{x_t\}_{t=1}^{m}$, $x_t \in \mathbb{R}^d$. By finding the closest codeword to x_t, we partition $\{x_t\}_{t=1}^{m}$ into K groups. In each group, we obtain vector v_i by accumulating differences between codeword u_i and its corresponding BIGD descriptors. The expression of v_i is shown as follows:

$$v_i = \sum_{x_t \in u_i} (u_i - x_t) \,. \tag{4.3}$$

Finally, by concatenating $\{v_i\}_{i=1}^{K}$, we obtain the encoded descriptor of an image with the length of dK.

IFV Encoding

Fisher encoding [94] uses Gaussian mixture models (GMM) to represent the distribution of local BIGD descriptors and captures the derivatives of GMM with respect to model parameters. Given prior probability π_k, mean u_k, and covariance matrix $\sum_k$, $k = \{1, 2, \cdots, K\}$, denoted $\Theta = \{\pi_k, \mu_k, \Sigma_k; k \in \{1, 2, \cdots, K\}\}$, the distribution of BIGD descriptor $x \in \mathbb{R}^d$ can be described by $p(x|\Theta) = \sum_{k=1}^{K} \pi_k p\left(x|u_k, \Sigma_k\right)$. To learn model parameters π_k, u_k, and $\sum_k$, we apply expectation maximization (EM) to BIGD descriptors extracted from the patches of training images. Then we calculate the derivatives of $\log p(x|\Theta)$ with respect to u_k and $\sum_k$ as follows:

$$\begin{cases} \dfrac{\partial \log p(x|\Theta)}{\partial \mu_k} = h_k \Sigma_k^{-1}(x - \mu_k) \\ \dfrac{\partial \log p(x|\Theta)}{\partial \Sigma_k^{-1}} = \dfrac{h_k}{2}\left(\Sigma_k - (x - \mu_k)^2\right) \end{cases}, \tag{4.4}$$

where $h_k = \frac{\pi_k p(x|u_k, \Sigma_k)}{\sum_k \pi_k p(x|u_k, \Sigma_k)}$. FV encoding concatenates all derivatives for the K components of GMM and obtains a vector with the length of $2dK$. The details of FV encoding can be found in [94], and in our experiments we use its improved version, IFV [95], which uses signed-square-root embedding followed by L_2 normalization.

4.2 Discriminative Multi-scale Texture Representation

4.2.1 Block Pair Formulation

Our developed BIGD describes the characteristic structures of patches that are evenly sampled with a step size of two pixels across the entire texture image and overlap with each other. The diagram that extracts the BIGD descriptor from a texture patch is shown in Fig. 4.3. To investigate the structural features of image patches, we randomly select multiple pairs of smaller square regions with various scales. Features extracted from these region pairs encode the local structures of patches at different spatial granularities and ori-

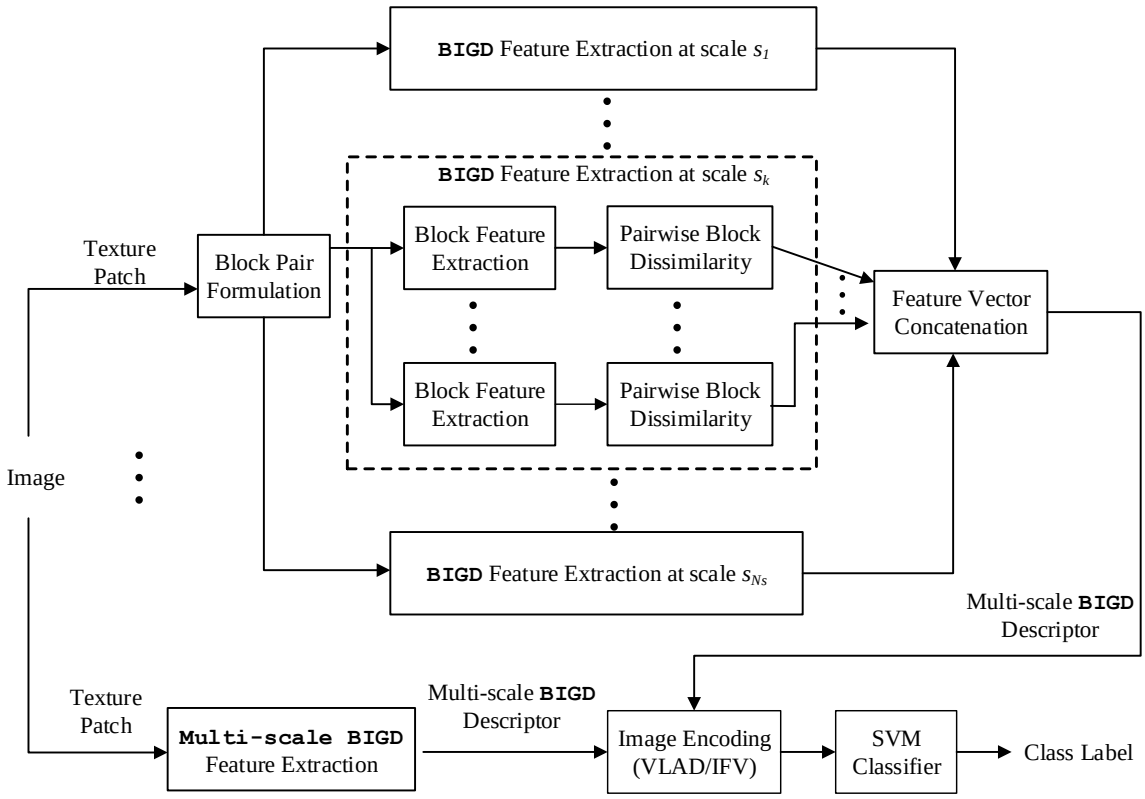


Figure 4.3: The diagram of extracting multi-scale BIGD descriptors from a texture patch.

entations and have higher robustness to noise than those extracted from raw pixels [64]. For simplicity, we specify these smaller square regions within the image patch as "blocks". An image patch of size 19×19 centered at C_p as an example in Fig. 4.4 contains block pairs connected by lines, where blue and red blocks have the sizes of 1×1 and 3×3, respectively. Only three block pairs at each scale are shown in Fig. 4.4, but in our experiments we consider a greater number of block pairs (e.g. 4 block pairs/scale) at more scales (e.g., 4 scales). We denote block pairs as $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, 2, \cdots, N$, where N defines the number of block pairs in the image patch. As Fig. 4.4 shows, $\mathbf{x}_i = [x_{i1}, x_{i2}]$ and $\mathbf{y}_i = [y_{i1}, y_{i2}]$ are the coordinates of the central pixels of the two blocks belonging to the i-th pair. Since blocks in the image patch are randomly selected, we identify the centers of all pairwise blocks using two sets of sampling points, $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and $Y = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N\}$. In the image patch, the coordinates of all block centers are represented by the coordinate system with the origin at patch center C_p. Following the sampling strategy in [96], we select sam-

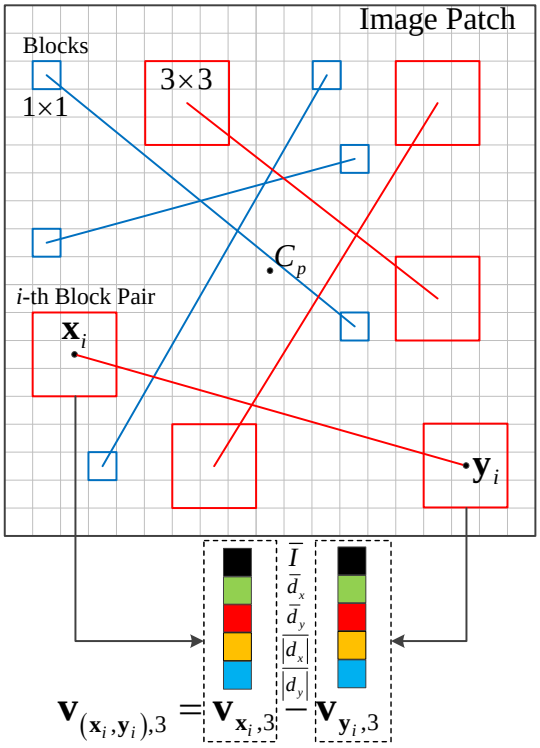


Figure 4.4: Block pairs with different scales in an image patch and the feature difference of pairwise blocks $(\mathbf{x}_i, \mathbf{y}_i)$ at scale 3.

pling points in X and Y from the isotropic Gaussian distribution, denoted $(X, Y) \sim$ i.i.d. Gaussian$(0, L^2/25)$, where L is the size of image patches. (X, Y) define the positions of N block pairs relative to an image patch. During the process of feature extraction, we will keep these block pairs unchanged for all patches over the entire image.

4.2.2 Multi-scale Block Intensity and Gradient Differences (BIGD)

BIGD Feature Extraction

By comparing the difference between pairwise blocks in various perspectives, we describe the local structures of patches. As introduced in [64], the average intensity difference of pairwise blocks captures variations in an image patch. Here, we denote the average intensity of blocks as $\bar{I}$. However, depending only on this feature, we cannot properly characterize the dissimilarity of pairwise blocks. Therefore, we utilize the average horizontal and vertical gradients of blocks, denoted $\bar{d}_x$ and $\bar{d}_y$, respectively, to capture smoothness.

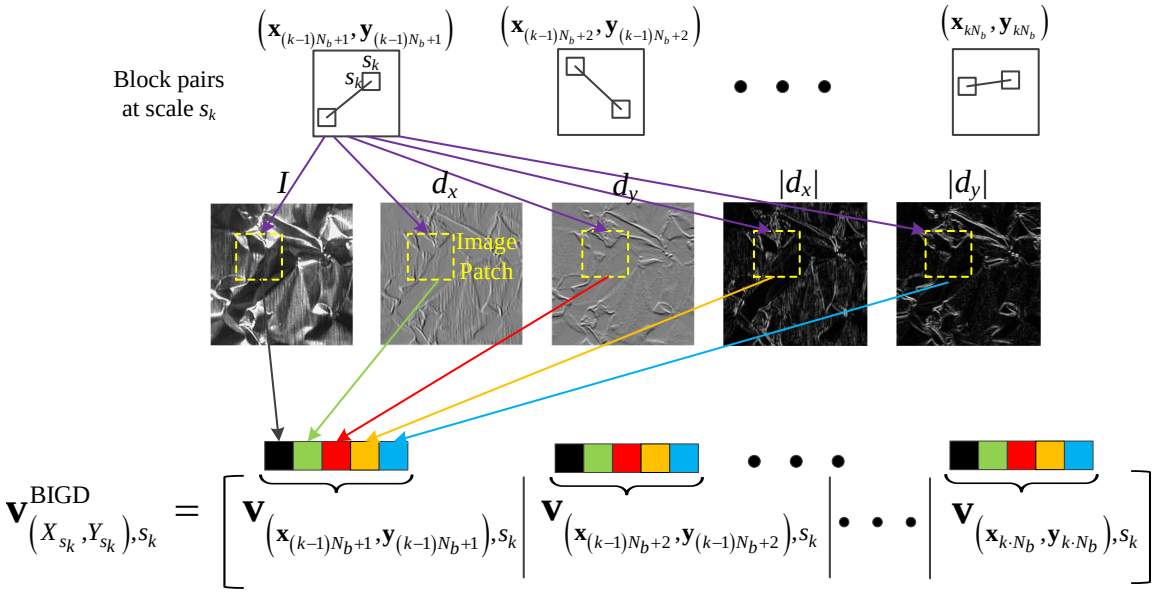
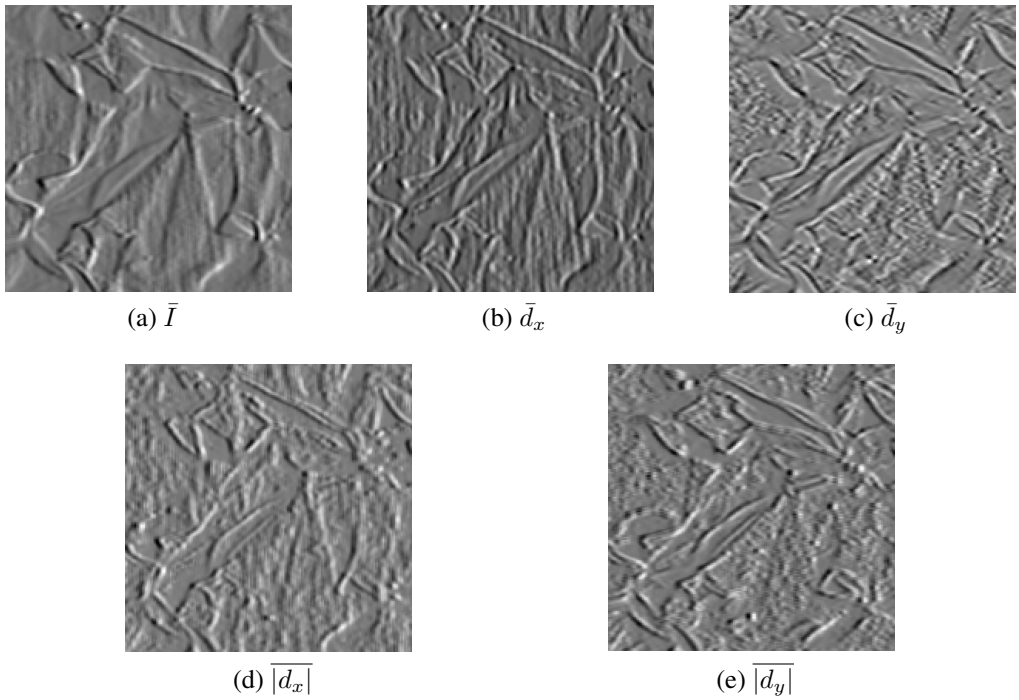


Figure 4.5: The extraction of the BIGD descriptor from an image patch using randomly sampled block pairs at scale s_k.

To obtain $\bar{d}_x$ and $\bar{d}_y$, we first apply the Sobel operator on all pixels in the patch and then average the horizontal and vertical gradients of pixels in blocks.

In addition, in order to analyze the polarity of intensity changes in patches, we average the absolute values of horizontal and vertical gradients and obtain another two features, denoted $\overline{|d_x|}$ and $\overline{|d_y|}$, respectively. Therefore, the block centered at $\mathbf{x}_i$ with scale s corresponds to a five-dimensional feature vector, denoted $\mathbf{v}_{\mathbf{x}_i,s} = \left(\bar{I}, \bar{d}_x, \bar{d}_y, \overline{|d_x|}, \overline{|d_y|}\right)$. The dissimilarity between pairwise blocks $(\mathbf{x}_i, \mathbf{y}_i)$ at scale s is evaluated by the difference between the feature vectors of corresponding blocks, denoted $\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),s} = \mathbf{v}_{\mathbf{x}_i,s} - \mathbf{v}_{\mathbf{y}_i,s}$. For clarity, we define $\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),s}$ as the feature vector of pairwise block $(\mathbf{x}_i, \mathbf{y}_i)$ at scale s. The bottom of Fig. 4.4 shows an example of calculating $\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),3}$ in a patch, where color squares represent different features. After extracting $\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),s}$ from all patches, we obtain five feature maps. Fig. 4.5 shows the process of extracting the BIGD descriptor of an image patch using randomly sampled block pairs at scale s_k, where the first row illustrates the random sampling strategy. The second row of Fig. 4.5 represents raw intensity and gradient maps, from which $\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),s_k}$ is calculated as shown in the third row of Fig. 4.5. In Fig. 4.6, we use

block pair $(\mathbf{x}_i, \mathbf{y}_i)$ at scale 3, where $\mathbf{x}_i = [-2, -5]$ and $\mathbf{y}_i = [-2, -1]$, to obtain five feature maps from raw intensity and gradient maps shown in Fig. 4.5. Notably these feature maps contain similar structures and that is because all features are extracted from the same block pair. However, different details in feature maps provide more information about the local structures of patches.



(a) $\bar{I}$ (b) $\bar{d}_x$ (c) $\bar{d}_y$

(d) $\overline{|d_x|}$ (e) $\overline{|d_y|}$

Figure 4.6: Five feature maps extracted by block pairs $([-2, -5], [-2, -1])$ at scale 3 from all patches of an image.

Multi-scale Extraction Scheme

The random selection of block pairs determines that extracted features can describe the local structure of patches in various orientations. To acquire a more discriminative representation of patches, we sample block pairs at multiple scales. We denote a set of scales as $S = \{s_1, s_2, \cdots, s_{N_s}\}$, where N_s represents the number of scales. Since an image patch contains N block pairs and we assume that every scale is of the same importance, the number of block pairs at each scale is $N_b = N/N_s$. We rewrite X and Y in Section 4.2.1 as

Table 4.1: Descriptions of five public texture databases.

Databases	# Classes	# Images	Image Size (pixels)	# Train /Class	# Test /Class	Capturing Conditions
Brodatz [29]	32	2048	200×200	32	32	16 samples, rotation- and scale- variations
CUReT [58]	61	5612	200×200	46	46	1 sample, various viewing angles and illuminants
KTH-TIPS [97]	10	810	200×200	40	41	1 sample, 3 viewing angles, 3 illuminants, and 9 scales
KTH-TIPS-2a [97]	11	4752	200×200	324	108	4 samples, 3 viewing angles, 4 illuminants, and 9 scales
KTH-TIPS-2b [97]	11	4752	200×200	324	108	4 samples, 3 viewing angles, 4 illuminants, and 9 scales

$X = \left[X_{s_1}, X_{s_2}, \cdots, X_{s_{N_s}}\right]$ and $Y = \left[Y_{s_1}, Y_{s_2}, \cdots, Y_{s_{N_s}}\right]$, respectively, to identify block pairs at different scales. (X_{s_k}, Y_{s_k}), $k = 1, 2, \cdots, N_s$, which contains the centers of pairwise blocks at scale s_k, can be expressed as follows:

$$(X_{s_k}, Y_{s_k}) = \left\{ \left(\mathbf{x}_{(k-1)N_b+1}, \mathbf{y}_{(k-1)N_b+1}\right), \left(\mathbf{x}_{(k-1)N_b+2}, \quad \mathbf{y}_{(k-1)N_b+2}\right), \cdots, \left(\mathbf{x}_{kN_b}, \mathbf{y}_{kN_b}\right) \right\}. \tag{4.5}$$

On the basis of (X_{s_k}, Y_{s_k}), in an image patch we calculate the features of pairwise blocks at scale s_k and concatenate feature vectors to generate the corresponding BIGD descriptor at scale s_k, denoted $\mathbf{v}^{\text{BIGD}}_{(X_{s_k}, Y_{s_k}), s_k}$, as follows:

$$\mathbf{v}^{\text{BIGD}}_{(X_{s_k}, Y_{s_k}), s_k} = \left[\mathbf{v}_{(\mathbf{x}_{(k-1)N_b+1}, \mathbf{y}_{(k-1)N_b+1}), s_k} \middle| \cdots \middle| \mathbf{v}_{(\mathbf{x}_{kN_b}, \mathbf{y}_{kN_b}), s_k} \right]. \tag{4.6}$$

Fig. 4.5 illustrates the process that extracts the BIGD descriptor at scale s_k. By concatenating BIGD descriptors at all scales, we obtain the BIGD descriptor at all scales, denoted $\mathbf{v}^{\text{BIGD}}_{(X,Y),S}$, which describes the local structures of an image patch at different granularities and orientations. The expression of $\mathbf{v}^{\text{BIGD}}_{(X,Y),S}$ is shown as follows:

$$\mathbf{v}^{\text{BIGD}}_{(X,Y),S} = \left[\mathbf{v}^{\text{BIGD}}_{(X_{s_1}, Y_{s_1}), s_1}, \cdots, \mathbf{v}^{\text{BIGD}}_{(X_{s_{N_s}}, Y_{s_{N_s}}), s_{N_s}} \right]. \tag{4.7}$$

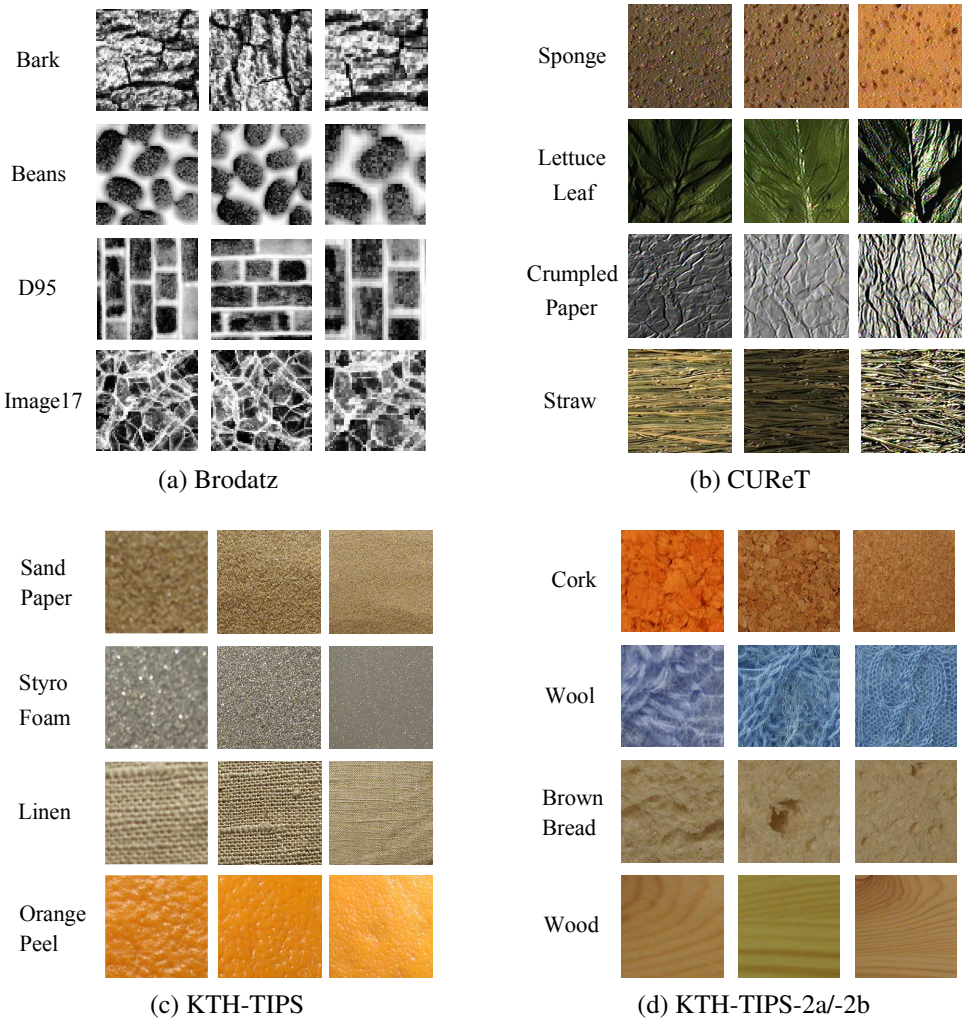


Figure 4.7: Five typical public databases for texture classification: Brodatz, CUReT, KTH-TIPS, and KTH-TIPS-2a/-2b.

4.3 Experimental Results in Texture Classification

4.3.1 Implementation and Parameter Effects

Implementation Details

The general pipeline of texture classification consists of three main modules: feature extraction, image encoding, and classification. In the first module, texture descriptors focus on describing the representative features of texture images. To obtain the BIGD descriptor that efficiently captures structural details of texture patches, first of all, we evenly sample

the centers of patches with a step size of two pixels across the entire texture image. Every sampled center corresponds to a local patch with a size of $L \times L = 15 \times 15$. For local patches, we apply a Gaussian random sampling strategy and select $N = 16$ block pairs. In our experiments, the local patches of all images in a database share the same layout of block pairs. To generate the more discriminative representations of patches, we extract BIGD descriptors in a multi-scale framework, where pairwise blocks have $N_s = 4$ scales ranging from 1×1 to 4×4. Under the assumption that each scale is of same importance, the number of block pairs at each scale is $N_b = N/N_s = 4$. From a block pair at one scale, we extract five features involving intensity and gradient differences. Therefore, by concatenating the feature vectors of block pairs at all scales, we obtain the multi-scale BIGD descriptor of a local patch with a dimension of $d = 5N_bN_s = 80$.

Image coding as the second module of texture classification encodes local BIGD descriptors into a full image descriptor using VLAD or IFV. An important step of image coding is to obtain model parameters trained on the local BIGD descriptors of training images. For example, the KTH-TIPS database have 400 (i.e., 40 training images/class $\times 10$ classes) training images with the size of 200×200. By sampling patch centers with a step of two in each training image, we identify $10,000$ patches. Rather than using all $4,000,000$ patches sampled from training images, in practical implementation we randomly select $500,000$ patches for computational efficiency. By training the local descriptors of selected patches, we obtain the codebook of VLAD and parameter set Θ of IFV. We set the number of clusters K as 128 in both VLAD and IFV for consistency and use the MATLAB® VLFeat toolbox [98] to implement VLAD and IFV encoding. To guarantee fair comparisons between our method and state-of-the-art ones, we keep the parameter setting unchanged for all databases unless we specify it.

In the SVM classification module, we randomly split each database into training and testing sets using testing protocols in Table 4.1 and repeat the partition ten times. In the tables of this paper, we use two metrics, the average and standard deviation of classifica-

tion accuracy over ten splits, to evaluate the performance of various descriptors on texture classification.

Database

In order to show the superiority of the BIGD descriptor over other state-of-the-art texture descriptors, we are going to evaluate their corresponding performance on texture classification. In this section, we conduct a set of experiments on five public texture databases: Brodatz [29], CUReT [58], KTH-TIPS [97], and KTH-TIPS-2a and -2b [97], in which texture images are captured under various conditions with the changes of occlusions, viewpoints, and illuminants. To illustrate the changes of capturing conditions, we randomly select four classes of textures from each database and exhibit three samples of every selected class in a row as Fig. 4.7 shows. Texture images from KTH-TIPS-2a and -2b are shown in Fig. 4.7(d) together since these two databases have the same texture classes.

Although we notice that some samples are color images, in our experiments we use only gray-scale images. To keep consistency and ensure fair comparison, all images are resized to 200×200. We give a brief description of each database in Table 4.1, which involves the number of classes, the number of total images, the image size, the number of images per class, the numbers of training and testing images per class, and capturing conditions. The combinations of various capturing conditions on image samples generate all texture images in each class. For example, although each class of the KTH-TIPS database has only one image sample, from this image sample the combinations of capturing conditions including three viewing angles, three illuminants, and nine scales generate $3 \times 3 \times 9 = 81$ images. In addition, the numbers of training and testing images per class determine the testing protocol of texture classification.

The original Brodatz database [99] consists of 32 classes of textures, each of which contains 16 image samples. By applying rotation, scaling, or both operations on original image samples, we obtain an extended database, in which each class contains $16 \times 4 = 64$

images. Following the testing protocol in [29], for each class, we randomly select half of images (i.e., 32 images/class) for training and use the remaining (i.e., 32 images/class) for testing.

In contrast to the extended Brodatz database, the CUReT database [58] is more challenging for texture classification. It consists of images acquired under various viewing angles and illuminants, which result in significant changes of texture appearances. The CUReT database is composed of 61 classes, each of which contains 92 images. The testing protocol in [67] requires us to randomly select half of images in each class (i.e., 46 images/class) for training and use the remaining (i.e., 46 images/class) for testing.

As an extension of the CUReT database, the KTH-TIPS [97] database selects a subset of images from the CUReT database and adds scale variations to these images. The KTH-TIPS database consists of ten classes, and each class contains 81 images captured under three viewing angles, three illuminants, and nine scales. According to the testing protocol in [67], in each class we randomly select half of images (i.e., 40 images/class) for training and use the remaining (i.e., 41 images/class) for testing.

As the two extensions of the KTH-TIPS database, KTH-TIPS-2a and -2b databases, which are designed for the recognition of surface materials, contain the images of 11 classes of materials such as wood and wool. In these two databases, each class consists of four physical samples, and each physical sample corresponds to 108 images captured under three viewing angles, four illuminants, and nine scales. Following the test protocol in [67], we use three physical samples of each class (i.e., 324 images/class) for training and the remaining one (i.e., 108 images/class) for testing.

Effects of Parameters

The performance of BIGD descriptors on texture classification depends on several factors such as patch sizes, block sizes, the number of k-means or GMM clusters, and the testing protocol. To understand the effects of these parameters on the performance of texture clas-

sification, we conduct our experiments mainly on Brodatz and KTH-TIPS-2a databases. These two databases are selected because of the great difference between their corresponding texture types.

For simplicity, experiments in this section sample block pairs at a fixed scale and utilize VLAD as the encoding method. In addition, testing protocols will follow Table 4.1 unless we specify it.

Patch and Block Sizes: According to our previous discussion, patch size $L \times L$ and block size $s \times s$ determine local BIGD descriptors. Therefore, we extract local BIGD descriptors with various parameter pairs (L, s) from Brodatz and KTH-TIPS-2a databases and list the corresponding classification results in Tables 4.2 and 4.3, respectively. In these two tables, we notice that if the block size is fixed, the increasing of the patch size improves average classification accuracy. The main reason is that patches with a larger size cover more local details and provide more choices of block pairs without involving redundant information, which comes from the overlapping of block pairs. In Table 4.2, for the Brodatz dataset, our method using parameter pair $(L, s) = (15, 3)$ achieves the highest classification accuracy 99.8%. In contrast, as Table 4.3 shows, for the KTH-TIPS-2a database, our method with parameter pair $(L, s) = (13, 2)$ has the classification accuracy of 83.23%, which achieves at most a 4.64% increase compared to other parameter pairs. The best choice of parameter pair (L, s) changes with databases, which implies that fixed patch and block sizes may not be able to accurately capture the details of textures. Therefore, to generate the more universal and discriminative representations of patches in different databases, we sample block pairs at multiple scales when extracting local BIGD descriptors. Coarse-level blocks reduce the effect of noise while fine-level blocks capture the details of local patterns. In addition, according to Sec. 4.2.2, the dimension of local BIGD descriptors keeps unchanged regardless of the scales of block pairs. Therefore, the comparison between classification accuracy in Tables 4.2 and 4.3 and that of our multi-scale BIGD descriptor is meaningful.

Numbers of k-means Clusters: K-means clusters describe the distribution of local fea-

Table 4.2: Classification accuracy of our BIGD method on the Brodatz database using different (L, s) pairs.

Block Size	Patch Size ($L \times L$)			
($s \times s$)	9×9	11×11	13×13	15×15
1×1	99.29 ± 0.24	99.59 ± 0.16	99.64 ± 0.25	99.71 ± 0.25
2×2	99.56 ± 0.30	99.60 ± 0.25	99.69 ± 0.22	99.78 ± 0.15
3×3	99.53 ± 0.22	99.63 ± 0.30	99.71 ± 0.19	$\mathbf{99.80} \pm 0.20$
4×4	99.55 ± 0.21	99.49 ± 0.27	99.70 ± 0.17	99.68 ± 0.29
5×5	99.32 ± 0.49	99.62 ± 0.28	99.61 ± 0.21	99.59 ± 0.26

Table 4.3: Classification accuracy of our BIGD method on the KTH-TIPS-2a database using different (L, s) pairs.

Block Size	Patch Size ($L \times L$)			
($s \times s$)	9×9	11×11	13×13	15×15
1×1	80.46 ± 3.48	79.71 ± 3.09	80.27 ± 2.85	82.86 ± 1.83
2×2	80.74 ± 3.92	79.91 ± 2.93	$\mathbf{83.23} \pm 3.64$	82.10 ± 2.80
3×3	80.24 ± 4.89	81.47 ± 3.68	81.53 ± 4.02	80.50 ± 3.52
4×4	79.16 ± 6.18	80.79 ± 3.65	79.21 ± 3.75	80.65 ± 4.44
5×5	78.59 ± 2.50	80.66 ± 2.65	81.52 ± 3.66	80.38 ± 3.54

ture descriptors and the number of clusters effect the encoding performance. To explore the effect of the number of k-means clusters, K, we select a variety of K values ranging from 16 to 128 and list their corresponding classification accuracy on Brodatz and KTH-TIPS-2a databases in Table 4.4. For consistency, we set parameter pair (L, s) for the Brodatz database as $(15, 3)$ and for the KTH-TIPS-2a database as $(13, 2)$, which correspond to the best classification performance in Tables 4.2 and 4.3. As we mentioned above, if the dimension of a local BIGD descriptor is d, a full image descriptor encoded by VLAD has a dimension of dK. A higher K value corresponds to a more abundant vocabularies but leads to the increasing of feature dimensionality. In Table 4.4, we notice that the classification accuracy of two datasets keeps growing with the increase of K. Although K with a value larger than 128 may correspond to higher classification accuracy, for computational efficiency, we set K as 128 for all experiments unless mentioned otherwise.

Testing Protocols: In addition to several factors mentioned above, the testing proto-

Table 4.4: Classification accuracy of our BIGD method on Brodatz and KTH-TIPS-2a databases using the different numbers of k-means clusters.

Cluster Numbers (K)	16	32	64	96	128
Brodatz	98.47 ± 0.42	99.41 ± 0.30	99.62 ± 0.22	99.69 ± 0.22	$\mathbf{99.80} \pm 0.20$
KTH-TIPS-2a	77.25 ± 3.26	81.51 ± 1.86	82.93 ± 3.75	80.54 ± 1.80	$\mathbf{83.23} \pm 3.64$

Table 4.5: Classification accuracy of our BIGD method on Brodatz and KTH-TIPS-2a databases using different testing protocols.

Training # vs. Testing #	$1:3$	$1:1$	$3:1$
Brodatz	98.46 ± 1.11	99.80 ± 0.20	99.80 ± 0.18
KTH-TIPS-2a	67.35 ± 2.57	76.39 ± 2.43	83.23 ± 3.64

col or the ratio between the numbers of training and testing images also has an effect on classification performance. The changes of classification accuracy under different testing protocols reflect the robustness of our method. Following the testing protocol in [64], we test our approach on Brodatz and KTH-TIPS-2a databases by choosing parameter pairs (L, s) same to Table 4.4 and setting the number of k-means clusters as 128. The classification results of three testing protocols on two databases are shown in Table 4.5. It is certain that more training images lead to higher classification accuracy and smaller standard deviations. In addition, we notice that even though the ratio between the numbers of training and testing images is $1:3$, our method is still able to achieve the classification accuracy of 98.46% for the Brodatz database and 67.35% for the KTH-TIPS-2a database. This supports our claim that the combination of BIGD descriptors and VLAD has strong potentials on the discrimination of texture images.

4.3.2 Overall Comparison

In order to compare the classification performance of our method with those of typical and state-of-the-art ones, we conduct our experiments on five public texture databases, Brodatz, CUReT, KTH-TIPS, and KTH-TIPS-2a and -2b by following standard testing protocols listed in Table 4.1. The parameter settings of experiments in this section follow

Table 4.6: Classification performance comparison between our BIGD method and other typical and state-of-the-art methods.

(a) Brodatz

Methods	Accuracy (%)
LBP	87.2
LQP [100]	96.9
WLD [29]	96.5
LHS [101]	99.3
SIFT + IFV [64][27]	97.6
SDMD + IFV [31]	99.7
DMD + IFV* [64]	99.8 ± 0.2
BIGD + IFV [65]	$\mathbf{99.9} \pm 0.1$
BIGD + VLAD [65]	99.7 ± 0.1

(b) CUReT

Methods	Accuracy (%)
MR8 [28]	93.5
BIF [102]	95.8
RP [92]	98.5
CLBP [67]	97.3
SIFT + IFV [64][27]	98.1
DMD + IFV* [64]	98.4 ± 0.7
BIGD + IFV [65]	$\mathbf{99.0} \pm 0.5$
BIGD + VLAD [65]	98.1 ± 0.9
FV-AlexNet [14][82]	98.4
FV-VGGM [14][82]	98.7
FV-VGGVD [14][82]	99.0
DeCAF [15]	97.9 ± 0.4
DeCAF+IFV [15]	99.8 ± 0.1

(c) KTH-TIPS

Methods	Accuracy (%)
SSELBP [63]	98.1
BIF [102]	98.5
SRP [30]	**99.3**
COV-LBPD [103]	98.0
WLD [29]	91.1
SIFT + IFV [64][27]	97.3
DMD + IFV* [64]	97.6 ± 1.6
BIGD + IFV [65]	98.8 ± 1.1
BIGD + VLAD [65]	$\mathbf{99.0} \pm 0.8$
DeCAF [15]	96.9 ± 0.9
DeCAF + IFV [15]	99.8 ± 0.2

(d) KTH-TIPS-2a

Methods	Accuracy (%)
MS-BIF [102]	71.6
LHS [101]	73.0
COV-LBPD [103]	74.9
scLBP [104]	78.4
NDV [80]	77.1
SIFT + IFV [64][27]	76.6
DMD + IFV* [64]	80.3 ± 6.1
BIGD + IFV [65]	$\mathbf{81.3} \pm 3.6$
BIGD + VLAD [65]	81.2 ± 2.5
DeCAF [15]	78.4 ± 2.0
DeCAF + IFV [15]	84.7 ± 1.5

(e) KTH-TIPS-2b

Methods	Accuracy (%)
MC_SBP [105]	71.6
CDL [106]	76.3
S_H-SVM [59]	80.1
Timofte [107]	66.3
DMD+IFV* [64]	76.2 ± 4.1
BIGD + IFV [65]	81.4 ± 3.1
BIGD + VLAD [65]	$\mathbf{82.7} \pm 4.5$
FV-AlexNet [14][82]	77.9
FV-VGGM [14][82]	79.9
FV-VGGVD [14][82]	88.2

implementation details in Sec. 4.3.1. Table 4.6 shows the classification accuracy of various methods on these databases, which come from either original or related publications. For some methods, because of the lack of standard deviations in corresponding original or related publications, we list only average classification accuracy. In addition, "$*$" means that we execute the source codes of original papers and obtain corresponding results. Since we encode multi-scale BIGD descriptors using VLAD or IFV in our work, our methods are represented by "BIGD+VLAD" and "BIGD+IFV" in Table 4.6.

Table 4.6(a) lists the classification accuracy of various methods on the Brodatz database, in which our method (BIGD+IFV) has the best performance. To verify the superiority of the multi-scale sampling strategy, we compare the classification result of our method (BIGD+VLAD) with those in Table 4.2. To fairly compare multi- and single-scale BIGD descriptors, we set their dimensionality to be the same. For example, we select 16 block pairs within a patch in Section 4.3.1, we select 16 block pairs within a patch; if we have four scales, the number of block pairs at each scale is four; and if we have only one single scale, the number of block pairs at this scale is 16. We notice that BIGD+VLAD with the average classification accuracy of 99.7% outperforms 70% single-scale BIGD descriptors in Table 4.2. In addition, the standard deviation of BIGD+VLAD, 0.14%, is smaller than those of all single-scale descriptors in Table 4.2, which supports our claim that compared to the single-scale strategy with the same dimension, the multi-scale strategy has a universal representation and yields robust classification performance. From Table 4.3 and Table 4.6(d) for KTH-TIPS-2a, we have the same observation about the superiority of the multi-scale strategy. For the CUReT database, the classification accuracy of BIGD+IFV is 0.5% higher than that of the second best method RP [92] as Table 4.6(b) shows.

In Table 4.6(c), we compare the classification performance of our method on the KTH-TIPS database with those of other typical and state-of-the-art approaches. Our method (BIGD+VLAD) achieves the second best average classification accuracy of 99.0%. In contrast, SRP [30] with the best classification performance on the KTH-TIPS database

involves rotation-invariant features.

Inspired by the randomly sampling strategy in BRIEF [96], BIGD achieves comparable performance to SRP because of randomly sampled block pairs, which describe patches at different scales and orientations and enhance the ability of discrimination on rotated textures.

We present the experimental results of various approaches on KTH-TIPS-2a and -2b databases in Tables 4.6(d) and 4.6(e), respectively. In contrast to the KTH-TIPS database, KTH-TIPS-2a and -2b databases are more challenging because of more image samples and more variations of illuminants. In each class of these two databases, every physical sample corresponds to a set of 108 images, and testing protocols split each class based on different physical samples rather than the random selection strategy in other databases. Therefore, without any knowledge of testing images, classification tasks on KTH-TIPS-2a and -2b databases become more challenging and suffer the significant decrease of classification accuracy. For the KTH-TIPS-2a database, our method (BIGD+IFV) outperforms DMD [64] by 1.0%. In addition, compared to most single-scale descriptors in Table 4.3, our method (BIGD+VLAD) has the higher classification accuracy and smaller standard deviation, which shows the necessity of the multi-scale sampling strategy. For the KTH-TIPS-2b database, a covariance descriptor, S_H-SVM [59], achieves the second best accuracy 80.1%, which is 2.6% less than the classification accuracy of our method (BIGD+VLAD). From the Table 4.6, we observe that the BIGD descriptor has more improvement on more challenging datasets like KTH-TIPS-2a and 2b which supports our claim that it captures the distinctive patterns of patches at different orientations and spatial granularitiy levels.

In contrast to the state-of-the-art descriptor DMD, the main contribution of our developed BIGD descriptor is to involve the gradient and absolute gradient difference of block pairs in a local patch, which improves the distinctiveness of descriptors. To evaluate the benefits of gradient differences on texture classification and guarantee fair comparisons between BIGD and DMD descriptors, the BIGD descriptor in our experiments is designed

to have the same dimension as DMD. As shown in Table 4.6, if both BIGD and DMD descriptors are encoded by IFV, the former yields $0.1\% \sim 5.1\%$ higher classification accuracy than the latter on five databases, where the highest performance improvement happens on the KTH-TIPS-2b database. If we use VLAD as the encoding method, the improvement of classification accuracy on the KTH-TIPS-2b database reaches 6.4%. It means that VLAD contributes only 1.3% improvement, which is much less than BIGD descriptors. SIFT [27] as the most typical local descriptor creates a histogram for each key point by partitioning gradient orientations into bins and involving gradient magnitudes as weights. However, quantized orientation features may inevitably result in the loss of information. Our BIGD descriptor extracts intensity- and gradient-difference features at multiple orientations without quantization, which retains the discriminative power of features. Last but not the least, the computation efficiency of BIGD is similar to that of DMD, around 100 times faster than SIFT, which has been evaluated in [15] so we do not repeat the details here.

Deep convolutional network-based features have shown their strong ability as a universal representation in classification or recognition tasks. We list several deep convolutional network-based approaches such as an effective texture descriptor FV-CNN proposed by Cimpoi et al. [14] in Table 4.6 and borrow their classification results from [82]. However, Cimopoi et al.'s CNN-based methods extracts and train features from color images, while our method only uses gray-scale images but still comparable. And as [82] shows, global CNN activations lack geometric invariance resulting in their robustness limitations for recognizing images with high variations. In addition, although deep features are obtained from pretrained AlexNet, the training process of AlexNet on the ImageNet database requires high computational cost. In contrast, the extraction of BIGD features does not need extra training steps. In addition, our hand-crafted approach is more interpretable. Since gradients are more resilient to photometric changes than intensities, the difference of gradients in BIGD describes the variations of gradients in a local patch and improves distinctiveness. Therefore, we still believe our BIGD descriptor is among state-of-the-art,

hand-crafted local descriptors.

4.4 Summary

In this chapter, we introduce a novel local texture representation method, block intensity and gradient difference (BIGD), which achieves great distinctiveness and computational efficiency. Compared with other algorithms mentioned above, our main contribution is efficiently captures un-quantized gradient difference features in BIGD. The gradient difference captures the variations of gradients in a local patch and improves distinctiveness. Descriptors such as SIFT utilize gradient-based features to capture the orientation information. However, quantized orientations in them result in information loss. Our BIGD method extracts intensity- and gradient-difference features at multi-orientations without quantization and retains the discriminative power of features. We have three major steps to evaluate the performance of BIGD on texure classification. First, we randomly select block pairs within an image patch. For each block, we extract a five-dimensional feature vector that contains the means of intensity, gradient, and absolute gradient values. Since we randomly sample pairwise blocks with multiple scales and orientations, the BIGD descriptor that concatenates the differences of feature vectors extracted from block pairs reveals underlying texture structures at different spatial granularities and orientations. Second, we encode the BIGD descriptors of local patches into an entire image representation using feature encoding modules such as VLAD [55] or IFV [94][95]. Third, the texture images are classified using a linear SVM. The superior performance of our approach was demonstrated by an extensive evaluation on public texture databases. In future work, we will improve the discriminative power of the BIGD descriptor on rotation variations and extend it to other computer vision tasks such as object recognition.

Table 4.7: Denotations for Chapter 4.

Denotations	Meaning	First Appeared
$\bar{I}$	the average intensity of an image patch	Sec. 4.1.1
d_x and d_y	first-order gradients of grid cells within the patch	Sec. 4.1.1
$t(a,b)$	a thresholding function	sec. 4.1.1
f	a function for extracting $\bar{I}$, d_x, or d_y from a grid cell	Sec. 4.1.1
s	a single resolution scale	Sec. 4.1.1
$p \times p$	image patch size	Sec. 4.1.1
$X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}, Y = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N\}$	two sets of sampling points	Sec. 4.1.1
$\mathbf{v}^{\text{BIGD}}_{(X,Y),S}$	BIGD descriptor	Sec. 4.1.2
$\{u_i\}_{i=1}^K, u_i \in \mathbb{R}^d$	cluster centers	Sec. 4.1.2
$\Theta = \{\pi_k, \mu_k, \Sigma_k; k \in \{1, 2, \cdots, K\}\}$	GMM distribution	Sec. 4.1.2
$(\mathbf{x}_i, \mathbf{y}_i)$	block pairs	Sec. 4.2
C_p	image patch center	Sec. 4.2
$\mathbf{v}_{\mathbf{x}_i,s} = \left(\bar{I}, \bar{d}_x, \bar{d}_y, \overline{\lvert d_x \rvert}, \overline{\lvert d_y \rvert}\right)$	a five-dimensional local feature vector	Sec. 4.2
$\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),s} = \mathbf{v}_{\mathbf{x}_i,s} - \mathbf{v}_{\mathbf{y}_i,s}$	the difference between the feature vectors of corresponding blocks	Sec. 4.2
$\mathbf{v}_{(\mathbf{x}_i,\mathbf{y}_i),s}$	the feature vector of pairwise block $(\mathbf{x}_i, \mathbf{y}_i)$ at scale s	Sec. 4.2
$S = \{s_1, s_2, \cdots, s_{N_s}\}$	a set of scales	Sec. 4.2
N_s	the number of scales	Sec. 4.2
N	the number of block pairs	Sec. 4.2
$N_b = N/N_s$	the number of block pairs at each scale	Sec. 4.2
(X_{s_k}, Y_{s_k})	the centers of pairwise blocks at scale s_k	Sec. 4.2
$\mathbf{v}^{\text{BIGD}}_{(X_{s_k},Y_{s_k}),s_k}$	the BIGD descriptor at scale s_k	Sec. 4.2
$\mathbf{v}^{\text{BIGD}}_{(X,Y),S}$	the BIGD descriptor at all scales	Sec. 4.2

CHAPTER 5

DEEP LEARNING-BASED TEXTURE REPRESENTATION IN MATERIAL CLASSIFICATION AND SURFACE CHARACTERIZATION

Recognizing textures and materials in real-world images has played an important role in object recognition and scene understanding. In Chapter 3, we detected repetitive lattices from near-regular textures and extracted representative features from the corresponding texture patterns. However, the appearance of materials may range from perfectly regular to purely stochastic. Characterizing apparent or latent properties (e.g. surface smoothness) of materials in a form of irregular textures is a crucial problem in several industry sectors. In this chapter, we explore computational material characterization, which moves a step further beyond material recognition. Few works exist assessing material surface characteristics because of two following challenges:

1. Data Availability: Obtaining sufficient training data with annotated textural information is time consuming and labor intensive. Therefore, annotated data are generally limited.

2. Material Appearance Variations: As we mentioned in previous chapters, materials "in the fine-grained scale" may have similar appearance although they belong into different subcategories. Because of small inter-class variations between subcategories, describing materials with subtle differences becomes more difficult.

To overcome these two challenges, we formulate the problem of material surface characterization as a fine-grained texture classification problem, and study the effectiveness of deep learning-based texture representation techniques in tackling the task. For this purpose, we build an imaging system with various settings on lighting conditions, zoom levels, material types, geometric variations, and touching directions. On the basis of this system, we

collect a new, large-scale challenging microscopic material surface dataset (CoMMonS), geared towards an automated fabric quality assessment mechanism in an intelligent manufacturing system. We then conduct a comprehensive evaluation of state-of-the-art deep learning-based methods for texture classification using CoMMonS. Additionally, we design a multi-level texture encoding and representation network (MuLTER), which simultaneously leverages low- and high-level features to maintain both texture details and spatial information in the texture representation. Our results show that, in comparison with the state-of-the-art deep texture descriptors, MuLTER yields higher accuracy not only on our CoMMonS dataset for material characterization, but also on established datasets such as MINC-2500 and GTOS-mobile for material recognition. Our dataset and source codes will be published online at `https://ghassanalregib.com/publications/` and `https://github.com/olivesgatech`, which will serve as a benchmark evaluating deep learning-based techniques for both material characterization and, more generally, fine-grained texture classification.

5.1 Deep Learning-based Texture Representation

As discussed in Sec. 2.1.2, a standard BOW pipeline includes four major components: local feature extraction, dictionary learning, feature encoding and pooling, and classification. Based on our comprehensive study of different visual representations in texture recognition, we summarize key modules for each component in Fig. 5.1 and explain them as below:

1. Local Feature Extraction: This module generates local texture descriptors and may involve hand-crafted methods such as local binary pattern (LBP) or dense scale-invariant feature transform (SIFT), pretrained filter banks, or pre-trained convolutional neural networks (CNN).

2. Dictionary Learning: This module creates a codebook (or distribution) of local texture descriptors using offline unsupervised dictionary learning methods, which mainly

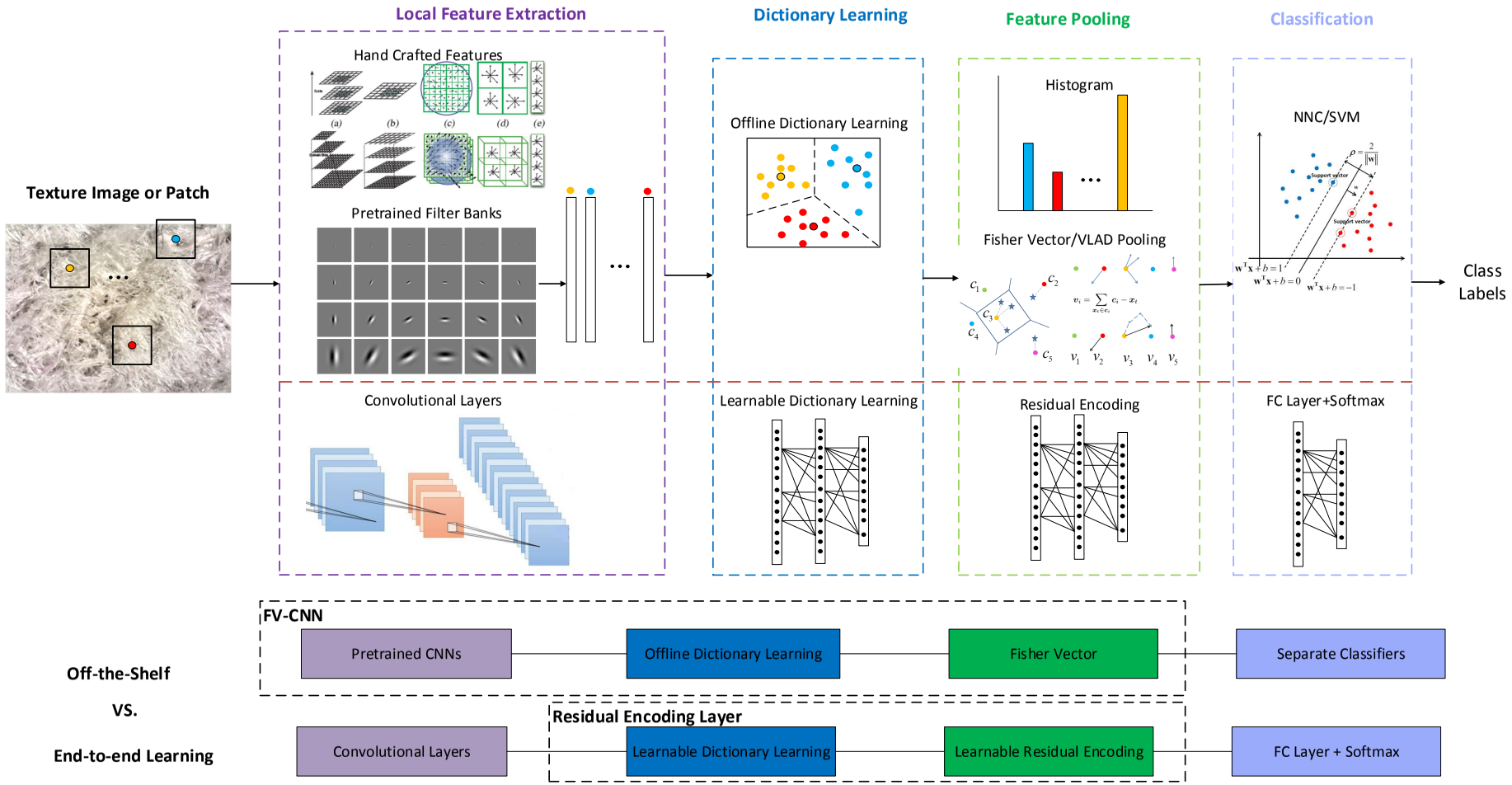


Figure 5.1: An overview of key modules in popular texture representation methods.

include k-means clustering and its probabilistic version, Gaussian mixture model (GMM). GMM contains K clusters, each of which is a Gaussian component with a mean, a variance, and a mixture weight.

3. Feature Encoding and Pooling: This module achieves orderless texture encoding using histograms, vectors of locally aggregated descriptors (VLAD), Fisher vectors (FV), or learnable residual encoding. In CNNs, a pooling layer (maximum or average pooling) are typically used on top of convolutional layers, and a global average pooling layer accumulates spatial information and generates a fixed-sized representation.

4. Classifiers: This module is commonly implemented by nearest neighbor classifier (NNC), support vector machine (SVM), or fully connected (FC) layer with softmax loss.

Different combinations of modules result in different approaches. There are two types of texture representation methods: BOW-based approaches and end-to-end learning based approaches, which are distinguished by whether the entire pipeline is jointly learned or not. A bag-of-words (BOW) based approach contains a pipeline, in which each major component is separately learned or optimized. In contrast, an end-to-end learning based approach represents the joint learning of all major components in an integrated manner. For the completeness of discussing learning-based methods in texture recognition, here we briefly describe two major prior work as shown at the bottom of Fig. 5.1, which are examples of BOW- and end-to-end learning based approaches, respectively. FV-CNN [14], the off-the-shelf representative in BOW-based methods, includes four modules, pretrarined CNN feature extractor, offline dictionary learning, Fisher vector, and separate classifiers. In contrast, end-to-end learning methods in [33, 16] jointly learn four modules, convolutional layers, dictionary learning, residual encoding, and the classifier constructed by a FC layer with softmax loss.

5.1.1 FV-CNN

FV-CNN [14] computes FV [34] on top of generic deep features (e.g., deep convolutional activation features (DeCAF) [108]). As Fig. 5.1 shows, FV-CNN contains three main modules, pretrained CNN feature extraction, offline dictionary learning by GMM, and FV feature pooling. To extract local features, FV-CNN utilizes a deep CNN pre-trained on the ImageNet dataset [32], which is commonly selected from VGG-M and VGG-VD networks. To recognize 1,000 object classes in the ImageNet dataset, the classification network alternates various layers including convolutional filtering, rectification, max pooling, normalization, and full linear weighting and learns their corresponding parameters. By removing the softmax and last fully-connected layer from the network, we obtain a feature extractor that generates deep features. The FV pooling of deep features is one of the best so far in texture and material recognition. As discussed in Sec. 4.1.2, the FV formulation is an improved BOW employing soft assignment weights. FV learns a soft codebook with a GMM including K modes, assigns each local feature to a codeword, and computes the gradient of the sample's likelihood with regard to GMM parameters. FV encodes higher order statistics (first and second order) aggregated residuals into a full image representation.

5.1.2 End-to-end Learning-based Texture Representation

One shortcoming of the FV-CNN architecture is the separation between CNN feature extraction, texture encoding, and classifier training, which does not benefit from the labeled data. As Fig. 5.1 shows, an end-to-end supervised learning method includes three major modules: convolutional layers, a learnable residual encoding layer, and a fully connected (FC) with a softmax layer, which are all differential with regards to a loss function.

Residual Encoding Layer

To jointly learn major components in the BOW pipeline together in an end-to-end manner, a texture encoding layer (i.e. a residual encoding layer) [33] builds the dictionary learning

and feature encoding on top of the CNN architecture and fuses them into one CNN layer referred as a residual encoding layer [16]. This deep texture encoding network learns an orderless representation through feature encoding, which performs well on texture/material recognition.

Here we introduce some notations. $X = \{x_1, \cdots, x_m, \cdots, x_M\}$ denotes M local texture descriptors, each of which includes dimension D. $C = \{c_1, \cdots, c_k, \cdots, c_K\}$ defines the codebook with K learnable codewords. The residual vector between x_i and c_j, where $i = 1, \cdots, m$ and $j = 1, \cdots, k$, is denoted as $r_{ij} = x_i - c_j$. The residual encoding regarding codework c_j, denoted e_j, is calculated by $e_j = \sum_i^K w_{ij} r_{ij}$, where w_{ij} represents the assigning weight for residual vector r_{ij}. The definition of w_{ij} is shown as follows:

$$w_{ij} = \frac{e^{-s_j||r_{ij}||^2}}{\sum_{n=1}^{m} -s_n||r_{in}||^2}, \tag{5.1}$$

where $s_1, \cdots, s_m$ are learnable smoothing factors. This encoding layer aggregate local descriptors X into K residual encoding vectors $E = \{e_1, \cdots, e_n, \cdots, e_N\}$. The number of codewords affects the representation capability of the encoding layer on textural information.

Bilinear Model

As textures or materials do not always exhibit completely orderless patterns, local spatial information is still useful for differentiating them. To resolve this issue, a deep encoding pooling network (DEP) with a bilinear model [109] jointly fuses the textural information and the spatial information. Bilinear model [109] processes their outputs as two independent factors. Provided the output feature vector from the texture encoding layer denoted x_1^d, and the vector from the global average pooling layer denoted x_2^d, the output of linear model is denoted as y^{d^2}. The bilinear function calculates their outer product and assigns learnable weights $w_{ij}, i = 1, \cdots, d, j = 1, \cdots, d$ to capture their pairwise correlation. The

function is given as follows:

$$y^{d^2} = \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_1^i x_2^j. \tag{5.2}$$

5.2 Multi-level Texture Encoding and Representation Network (MuLTER)

In this section, we design a multi-level texture encoding and representation network (MuLTER) for texture-related applications, whose architecture is shown in Fig. 5.2 and Table 5.1. We build the MuLTER on top of convolutional and non-linear layers (e.g., ResNet18 [110]) pretrained on ImageNet [52] and incorporate learnable encoding modules after the last layer of each ResNet block. As the architecture shown in Table 5.1, four levels of information are extracted from Res1, Res2, Res3, and Res4 blocks. Based on a multi-level pooling architecture, the MuLTER network simultaneously leverages low- and high-level features to maintain both texture details and spatial information. Such a pooling architecture involves few extra parameters and keeps feature dimensions fixed despite changes in image sizes. In comparison with state-of-the-art texture descriptors, the MuLTER network yields higher recognition accuracy on existing texture datasets such as MINC-2500 and GTOS-mobile with a discriminative and compact representation. In addition, we analyze the impact of combining features from different levels, which supports our claim that the fusion of multi-level features efficiently enhances recognition performance. Our source code will be published on `https://github.com/olivesgatech`.

5.2.1 Learnable Encoding Module (LEM)

We refer to the entire module in this part as a learnable encoding module (LEM), shown in Fig. 5.2 and Table 5.2. Notations in this section are introduced as follows. The input size of a LEM is $W \times H \times D$, where W, H, and D denote the width, the height, and the feature channel dimension of the input volume, respectively. In addition, the number of codewords

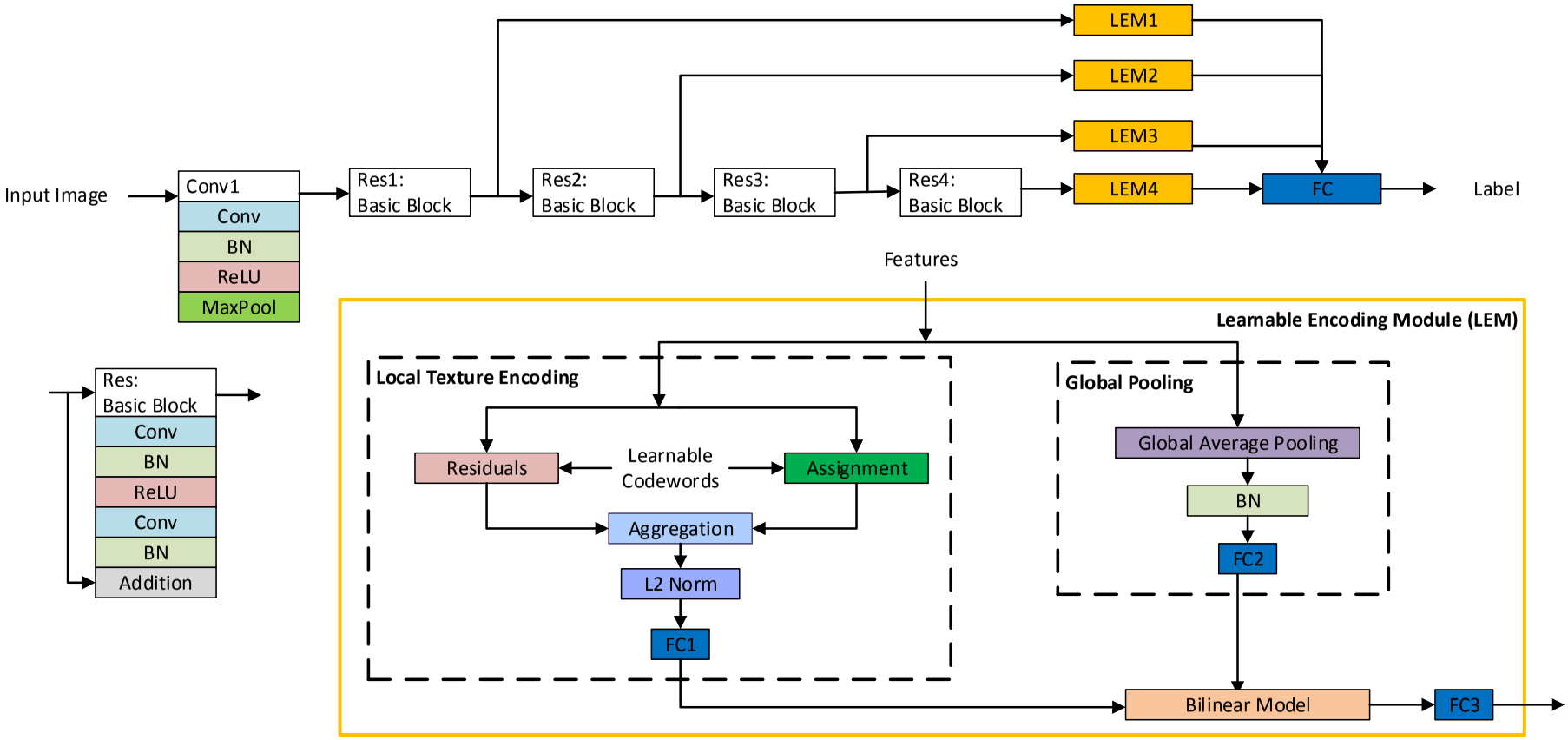


Figure 5.2: Flowchart of MuLTER for texture represetation.

Table 5.1: Architecture for adopting pretrained ResNet18.

Modules	Layers	Basic Blocks/Layers	Ouput Size	Multi-levels	LEM Output Size
	Conv1	7×7, 64, stride 2	112×112×64		
ResNet18	Res1	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	56×56×64	LEM1	C=128
	Res2	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	28×28×128	LEM2	C=128
	Res3	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	14×14×256	LEM3	C=128
	Res4	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	7×7×512	LEM4	C=128
Classifier	FC	128×4 = 512 =>n	n classes		

Table 5.2: Learnable encoding module (LEM). The 3rd column shows the output sizes for an input image size of $224 \times 224 \times 3$ and the 4th column shows the basic blocks or layers used.

Spatial	Layers	Output size	Basic Blocks/Layers
	Reshape	WH×D	W×H×D => WH×D
Textural Information	Encoding	K×D	K codewords
	Projection	64	FC1: KD =>64
Global Information	Pooling	D	Average Pooling
	Projection	64	FC2: 512 => 64
Textural & Global	Bilinear Model	4096	=> 64^2
	Projection	C=128	FC3: 4096 =>128

in the learnable dictionary is K.

An end-to-end deep learning-based framework for texture recognition generally involved a "local texture encoding" layer [33] shown in Fig. 5.2. This special layer jointly learns an inherent dictionary of local texture descriptors extracted from CNNs and generalizes robust residual encoders such as VLAD [55] and Fisher Vector [94] through a "residual" layer calculated by pairwise difference between texture descriptors and the codewords of the dictionary. In "assignment" layer, assignment weights are calculated based on pairwise distance between texture descriptors and codewords and the "aggregation" layer converts the residuals vectors and the assignment weights into a full image representation. Because of the residual encoding, such image representations discarding frequently appearing features are helpful to domain transfer learning. In addition to orderless texture details

captured by the encoding layer, local spatial information is important visual cues, and the "global pooling" layer [16] extracts global scene context by average pooling of local texture descriptors. Then, a bilinear model [111] follows the texture encoding layer and the global pooling layer to jointly combine the two types of complementary information.

5.2.2 Multi-level Deep Feature Fusion

The multi-level feature fusion means the joint utilization of both low-level features and high-level features from Res1 to Res4 of ResNet18. ResNet18 uses 4 basic blocks of similar structures and one example of the basic block is shown in the left bottom of Fig. 5.2. Given an input image with size $224 \times 224 \times 3$, after employing convolutional filters (i.e. Conv1, a default structure at the beginning of the Resnet family), the output size is $112 \times 112 \times 64$. Then, we feed it into ResNet18. Here, we have four levels, Res1, Res2, Res3, and Res4. The outputs from each level have different output sizes so we feed them into different sizes of LEMs. For example, for the first level, Res1 is followed by LEM1, where the output size of Res1 is $W \times H \times D = 112 \times 112 \times 64$ and LEM1 converts it into a feature vector of dimension $C = 128$. Whatever the input image size is, the same architecture shown in Table 5.1 can be used to produce a fixed-length (i.e., C) feature representation. Similar to the first level, we can repeat the procedure above to calculate a feature vector of dimension $C = 128$ for level 2, 3, and 4. For local CNN-based texture descriptors at each level with either low-level features or high-level features, we preserve both texture details and local spatial information through their corresponding LEMs. To combine the features from different levels, we concatenate them and feed them into a classification layer. Assuming the number of classes is n, the classification layer maps the $4C$ feature vector to n classes.

The multi-level architecture for texture encoding and representation has multiple advantages. First, the multi-level architecture makes it easy to adjust regarding which level of information should be fused. Second, it can be easily extended to other CNN models (e.g. ResNet50) by adapting the size of LEMs and the number of levels. Third, all mod-

ules in the overall architecture are differentiable, so the network can be trained with back propagation in an end-to-end texture encoding and representation network. Last but not the least, this architecture produces a compact yet discriminative representation with a full image representation with a dimension of a few hundreds (e.g. 512).

5.3 MuLTER in Material Classification

5.3.1 Datasets and Implementation Details

Datasets: To show the recognition performance of MuLTER method as a general texture representation technique for texture/material recognition, we test it on two recent challenging texture/material datasets: materials in context database (MINC)-2500 [50] and ground terrain database (GTOS)-mobile [16]. The MINC dataset is an order of magnitude larger than previous texture and material datasets (such as KTH-TIPS [47] and FMD [36]), while being more diverse and well-sampled across its 23 categories. For a fair comparison with other methods, we use MINC-2500 (i.e. a subset of MINC with 2500 patches per category). GTOS-mobile is a dataset including images for ground terrain regions captured by mobile phones. It consists of 31 classes such as grass, brick, soil, etc., and can be used for material classification. The GTOS-mobile is challenging because of its realistic capturing conditions (i.e. a mobile imaging device, handheld video, and uncalibrated capture). Compared with GTOS-mobile, MINC-2500 is a more general one.

Implementations: Following the standard testing protocol of MINC-2500 and GTOS-mobile, we use the same data argumentation and training procedure. We resize images to 256×256 and randomly crop patches to 224×224. For the training part, we augment data using horizontal flips with a 50% probability. For a fair comparison with [16], we build a ResNet18 for the GTOS-mobile dataset and build a ResNet50 for the MINC-2500 dataset. As mentioned in Sec. 5.2.2, our method is easily extended to other CNN models (e.g. ResNet50) by adapting the size of LEMs. Our experimental settings are: learning rate starting at 0.01 and decaying every 10 epochs by a factor of 0.1, batch size 128 for GTOS-

Table 5.3: Comparison of various level-selection schemes of MuLTER method on the MINC-2500 and the GTOS-mobile datasets.

Schemes	MINC-2500 [50]	GTOS-mobile [16]
L=1	59.10%	62.35%
L=2	70.84%	74.94%
L=3	80.70%	76.43%
L=4	81.01%	77.04%
L=1,2	70.16%	77.68%
L=3,4	81.29%	76.08%
L=1,4	81.29%	77.88%
L=1,2,3	80.45%	75.44%
L=2,3,4	81.44%	76.46%
L=1,2,3,4	82.21%	78.21%

mobile and 32 for MINC-2500, momentum 0.9, and the total number of epochs 30. The number of codewords K is set to 8 for GTOS-mobile and 32 for MINC-2500. The result is shown in Table 5.4, which shows the superior recognition accuracy of our multi-level architecture. We run experiments on a PC (Nvidia GeForce GTX1070, RAM: 8GB).

5.3.2 Experimental Results

Impact of Level Selections: Table 5.3 shows the results obtained from various schemes of level selection on MINC-2500 and GTOS-mobile, separately. Each scheme utilizes CNN features of different levels, from single levels (e.g. L=1 or L=4) to multiple levels (e.g. L=1,4 or L=1,2,3,4). Table 5.3 does not include an exhaustive comparison of different schemes since we skip those similar results and show the representative ones. From Table 5.3, we have several observations:

(1) For single levels on both datasets, the results under setting "L=4" outperforms setting "L=1". This indicates that with the CNN architectures, high-level features tend to describe textures better than low-level features. Setting "L=4" performs 21.91% better than "L=1" on the MINC-2500 dataset and 14.69% on the GTOS-mobile dataset.

(2) The setting "L=1,2,3,4" with four levels achieves the highest recognition accuracy

on the MINC-2500 dataset, and yields the highest accuracy on GTOS-mobile dataset. This observation supports our claim that fusing information from multiple levels improves the discriminative capability to better describe and differentiate various texture images.

(3) The benefits from multi-level feature fusion vary among datasets. For example, on the GTOS-mobile dataset, the second highest recognition accuracy obtained from setting "L=1,4" is just 0.33% lower than that of setting "L=1,2,3,4", which implies that "L=1,4" already captures sufficiently discriminative features, while features from"L=2,3" bring limited improvement. In contrast, on the MINC-2500 dataset, the second highest recognition accuracy comes from setting "L=2,3,4", 0.15% higher than that of setting "L=1,4" without medium-level features. Thus, features from setting"L=1" or "L=2,3" have comparable contributions to the improvement of the discriminative capability, but which one brings more improvement depends on specific datasets.

(4) The impact of incorporating features from certain levels can vary significantly among different datasets. For example, on the GTOS-mobile dataset, setting "L=1,2" already yields good performance and outperforms that from setting "L=3,4" while the MINC-2500 datset presents the opposite results. This indicates that for a larger, more diverse dataset like MINC-2500, features from deeper layers bring more improvement than those from shallow layers.

Comparison with State-of-the-art Methods: We evaluated our method and compared with other state-of-the-art methods on these two datasets, as shown in Table 5.4. The results for ResNet [110], FV-CNN [15], and Deep-TEN [33] were borrowed from [16]. The results for DEP were generated using codes [112] provided by the authors. On the MINC-2500 dataset, our method achieved a recognition accuracy of 82.2%, which outperforms Deep-TEN by 1.8% and DEP by 1.2%. On the GTOS-mobile dataset, the recognition accuracy of our method is 78.2%, which is 4.0% better than Deep-TEN and 1.2% better than DEP. The reason behind our enhanced performance is that our method fuses multi-level CNN features in a distinctive and compact way while Deep-TEN and DEP only use features from a single

Table 5.4: Comparison with state-of-the-art algorithms on the MINC-2500 and the GTOS-mobile datasets in %.

Methods	MINC-2500 [50]	GTOS-mobile [16]
ResNet [14]	N/A	70.8
FV-CNN [14]	63.1	N/A
Deep-TEN [33]	80.4	74.2
DEP [16]	81.0	77.0
MuLTER [113]	**82.2**	**78.2**

level.

In summary, we designed a multi-level deep architecture (MuLTER) in this thesis. It fulfilled a multi-level texture representation, simultaneously extracting low-level and high-level CNN features to maintain texture details and local spatial information. In comparison with the state-of-the-art techniques, MuLTER has accomplished higher recognition accuracy with a compact feature representation on two challenging texture datasets. Additionally, we analyzed the impact of incorporating CNN features from different levels on our method.

5.4 MuLTER in Material Surface Characterization

5.4.1 Material Surface Dataset Acquisition

Image Acquisition System

Imaging System: A carefully designed imaging system is critical for the analysis of fabric surface images. Such an imaging system needs to be powerful enough to capture the fine details of fabric surfaces, which reveal the textural differences that characterize the fabrics. While a regular digital camera usually cannot accomplish this task, a commercial imager, Dino-lite AM73915MZT is appropriate for our imaging system. Some key features of this microscope include: an optical magnification power ranging from 10 to 220; a high resolution of 2560$\times$1920 pixels; an automatic magnification reading (AMR) function that enables automatic magnification rate recording; an extended depth of field (EDOF) that

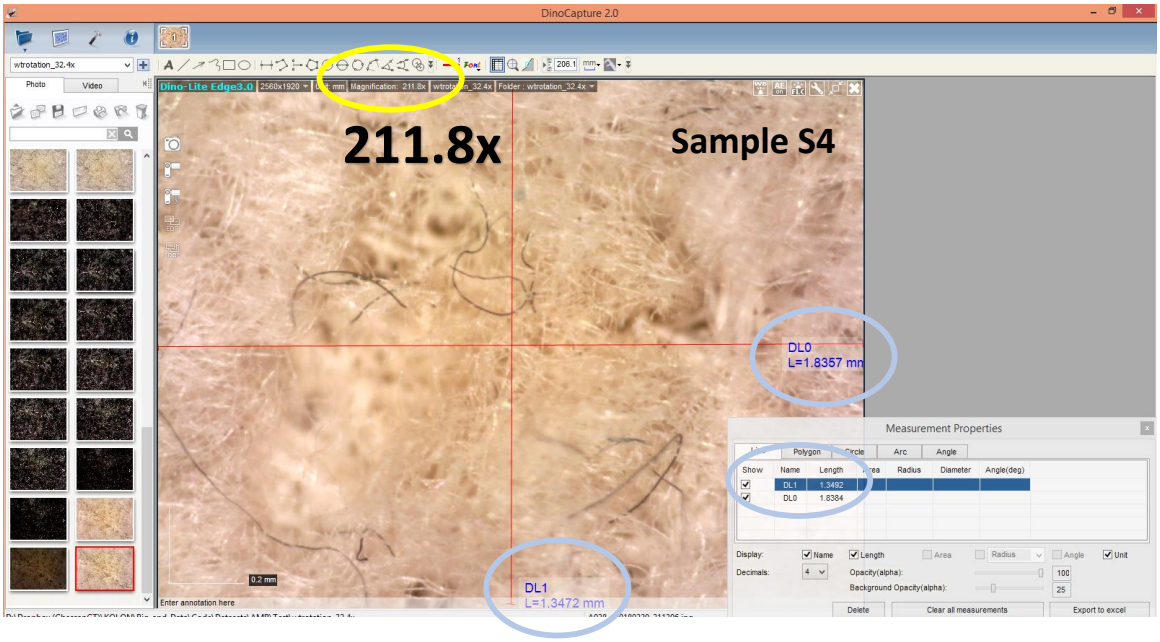


Figure 5.3: An commercial imager (Dino-lite AM73915MZT) and its interface.

provides enhanced image quality at high magnification rates; and an enhanced dynamic range (EDR) that provides enhanced image quality for limited dynamic range conditions. One example of the imager for sample "S4" is shown in Fig. 5.3, where the magnification rate, 211.8 is automatically read and the physical size of the image shows in the interface after only one-time imager calibration. The example images for EDR and EDOF are shown in Fig. 5.4.

Controlled Environment: For a dataset to be useful, it should be a comprehensive collection, representing different types of fabrics imaged in various conditions. Such a comprehensive coverage is crucial for validating the system and the algorithms, ensuring a robust performance. Therefore, the data acquisition system needs to be established within a controlled environment. It also has to incorporate various environmental conditions as encountered in real-world settings, such as variations in lighting, zoom-in level of the camera lens, position of the fabric, etc. Our main focus is to establish a staging system, which is the most challenging component of such a controlled environment. A motorized system is ideal for such large number of combinations. However, the cost of such a system

makes it difficult. Instead, we used commercial manual staging systems that would satisfy our requirements in terms of accuracy, repeatability, and efficiency. The complete data acquisition system is shown in Fig. 5.5.

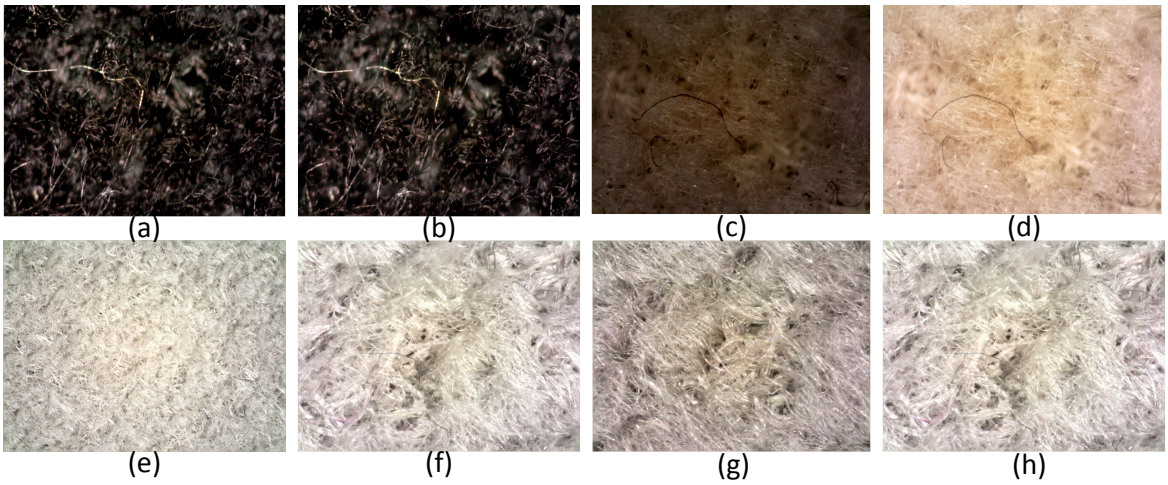


Figure 5.4: Example images obtained with the data acquisition system: (a) sample "S2" without EDOF; (b) sample "S2" with EDOF, which significantly reduces the blurry areas; (c) sample "S4" without EDR; (d) sample "S4" with EDR, which enhances the image details not revealed clearly under the original condition; (e) sample "S1" with camera zoom level 50; (f) sample "S1" with camera zoom level 200; (g) sample "S1" along the "pile" pressing direction; and (h) sample "S1" along the opposite "pile" pressing direction.

CoMMonS Dataset Collection

We created a comprehensive dataset of images captured at the fabric surfaces under varying conditions. We have 24 samples from "S1" to "S24" with subjective quality evaluation for three fabric properties. We acquired images for these samples under varying sample conditions and imaging settings as follows:

1. Three surface attributes: fiber length, smoothness, and toweling effect of material surfaces;

2. Six translation positions: six non-overlapping locations from each fabric sample for simulating translation variations by adjusting the translation staging, which enables the 6-fold cross-validation;

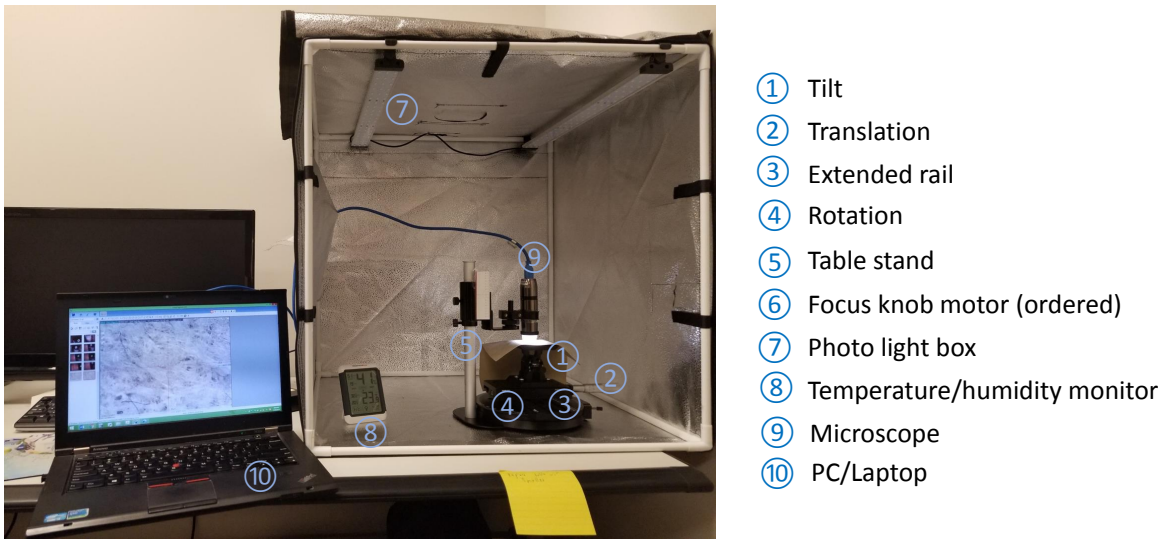


Figure 5.5: An illustration of the data acquisition system. The microscope is mounted on a table stand and connected to a laptop. A fabric sample is placed right under the microscope and on top of a manual staging system. The system (except for the laptop) is set up within a photo light box to ensure controlled lighting conditions. Controllable lighting sources are available with both the light box and the microscope. A temperature/humidity monitor is placed inside the light box to keep record of the temperature and humidity while acquiring images.

3. Two rotation angles: Two perpendicular directions noted R-30 and R+60 to simulate rotation variations by adjusting the rotation staging;

4. Two lighting conditions: Two appropriate lighting conditions among 6 options in the microscope by adjusting the imager software interface;

5. Two camera zoom-in levels: zoom 50 and zoom 200 by moving the distance between the microscope and a material surface;

6. Three camera function settings: Normal, EDOF, and EDR by adjusting the imager software interface;

7. Two sample conditions regarding touching (or pressing) directions: the along pile direction and the opposite direction of human pressure; the pressing directions are included because they will affect a human expert in evaluating the fabric properties.

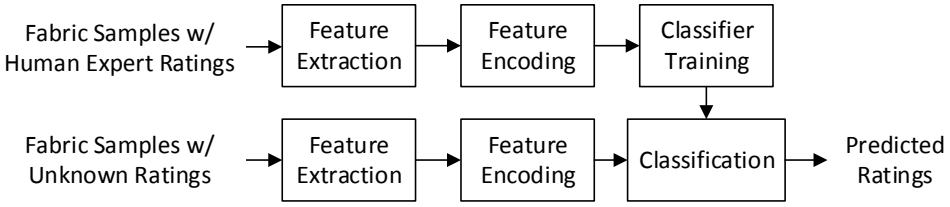
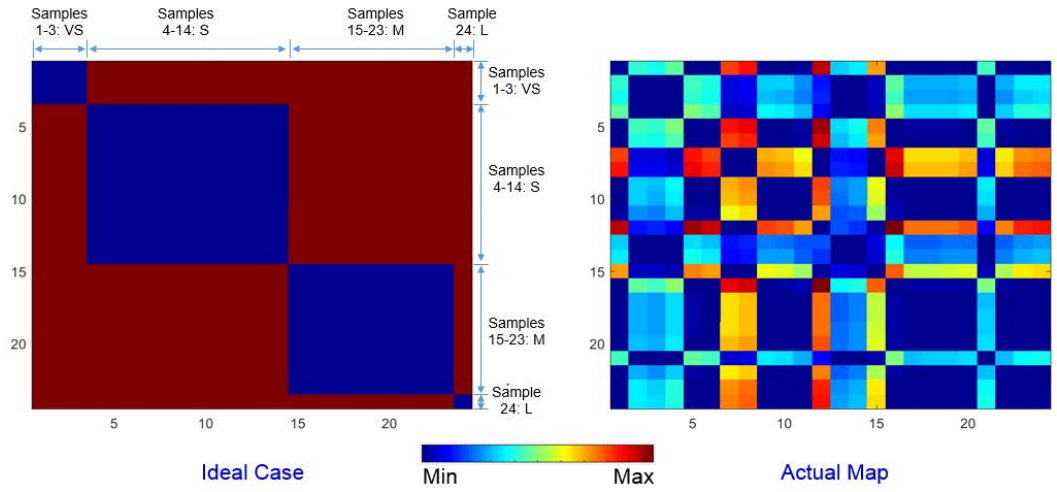


Figure 5.6: Supervised learning for material surface characterization.

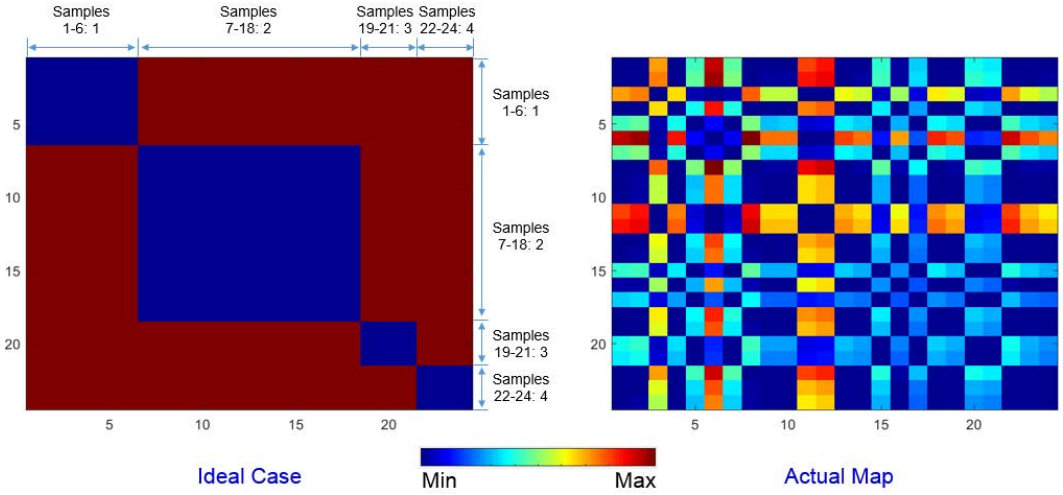
Combining all these conditions, the total number of images in this dataset is $24 \times 6 \times 2 \times 2 \times 2 \times 3 \times 2 = 6912$ high resolution images. Example images are shown in Fig. 5.4. We list key features of CoMMonS dataset and compare it with other counterparts in Table 2.1 of Sec.2.1.1. Different from other datasets, our dataset focuses on material characterization for one material (fabric) in terms of one of three properties (fiber length, smoothness, and toweling effect), facilitating a fine-grained texture classification. In this particular case, the dataset is used for a standard supervised problem of material quality evaluation, as shown in Fig. 5.6. It takes fabric samples with human expert ratings as training inputs, and takes fabric samples without human subject ratings as testing inputs to predict quality ratings of the testing samples. The texture patches are classified into 4 classes according to each surface property measured by human sense of touch. For example, the human expert rates surface fiber length into 4 levels, from 1 (very short) to 4 (long), and similarly for smoothness and toweling effect. Because the samples all belong to the same type of fabric, the intra-class appearance variation is much smaller, making the classification much more challenging. Also, our images are of much higher resolution comparing to those from other datasets.

5.4.2 Material Surface Image Analysis with Baseline Methods

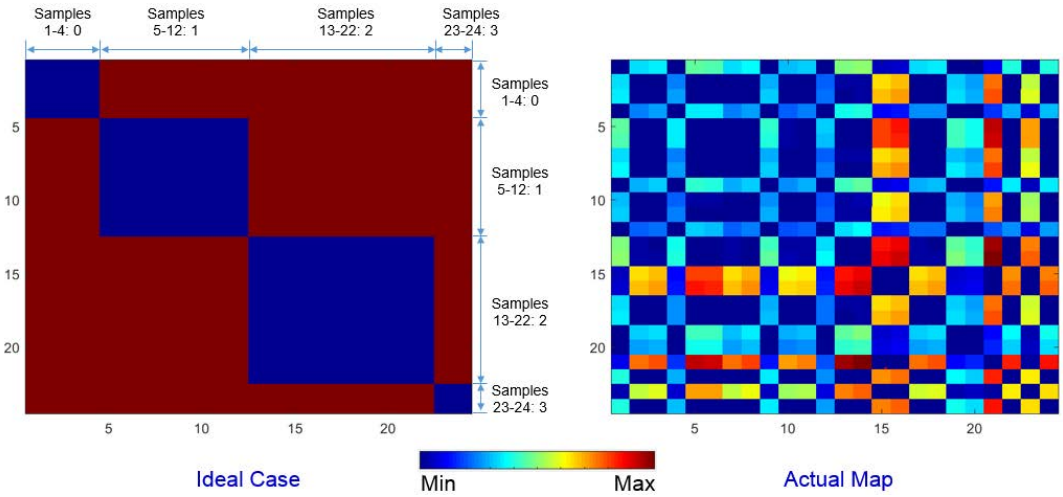
Before exploring deep learning methods, we evaluate traditional image analysis methods on material surface characterization including Tamura [38] perceptual method and binary descriptors such as local binary pattern (LBP) [26], completed local binary pattern (CLBP) [69], multi-scale completed local binary pattern (M-CLBP) [69], completed local



(a) Sample comparison using Tamura features regarding the "fiber length" property.



(b) Sample comparison using Tamura features regarding the "smoothness" property.



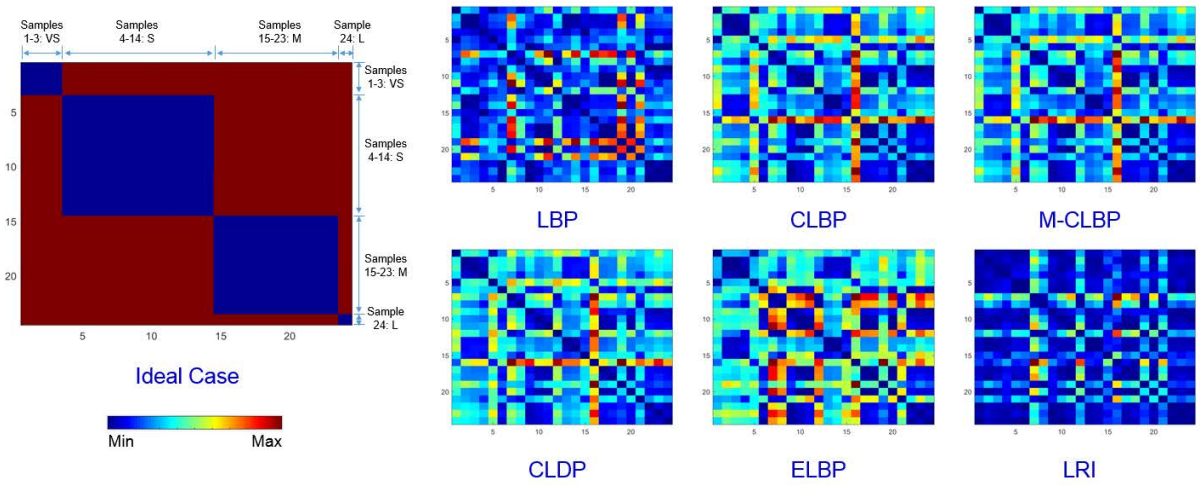
(c) Sample comparison using Tamura features regarding the "toweling effect" property.

Figure 5.7: Comparison of samples using the combination of the six baseline features for each individual fabric property. Again, the combined features cannot match the human ratings.

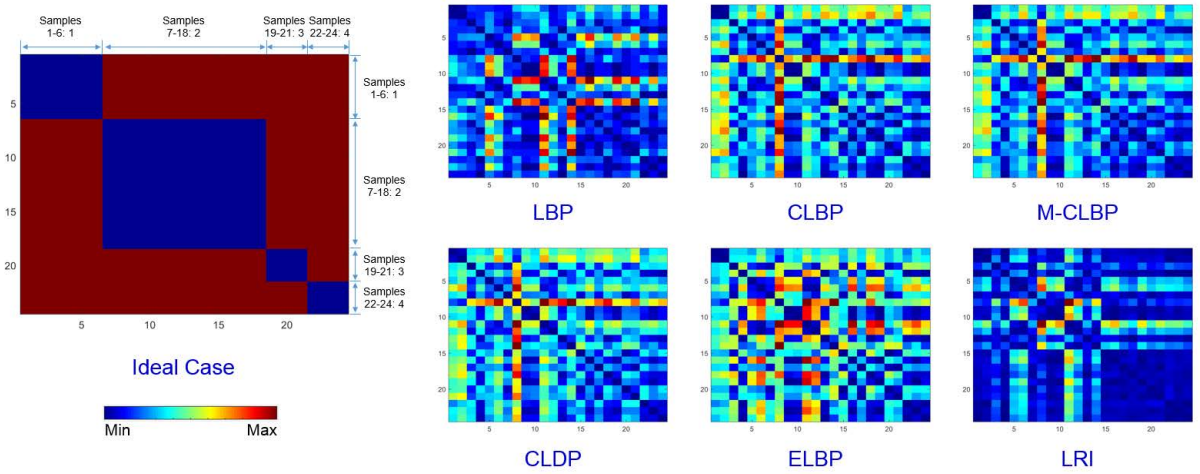
derivative pattern (CLDP) [60], extended local binary pattern (ELBP) [72], and local radius index (LRI) [114]. The binary descriptors are effective and efficient for texture image analysis. They examine textural patterns in both local and global areas. Computation is efficient because of the binary coding incorporated. Features extracted by Tamura or binary descriptors are compared between different samples to examine how effective they are for the task of differentiating between the samples.

Tamura Features: First, we implemented code that generates six baseline features, i.e., coarseness, directionality, line-likeness, contrast, roughness, and regularity. These baseline features are correlated with visual perceptions and are defined based on a psychological measurements for human subjects. We then examined these features using images in our dataset. We compared all samples by combining six features into one feature vector and calculating feature distance. We group samples with regards to a certain characteristics (fiber length, smoothness, or toweling effect) and establish a distance map, where each each entry gives a difference value presented in color code in associated features between the two corresponding samples. The results of the ideal distance map and the corresponding actual map are shown in Fig. 5.7a, Fig. 5.7b, and Fig. 5.7c regarding three surface texture attributes (i.e., fiber length, smoothness, and toweling), respectively. In the ideal case, difference is minimum between samples of the same type of characteristic, and is maximum between samples of different types. As we observe, the idea distance map the actual map look very different to each other. Therefore, Tamura features failed to match human expert ratings very well, and more advanced features are in need.

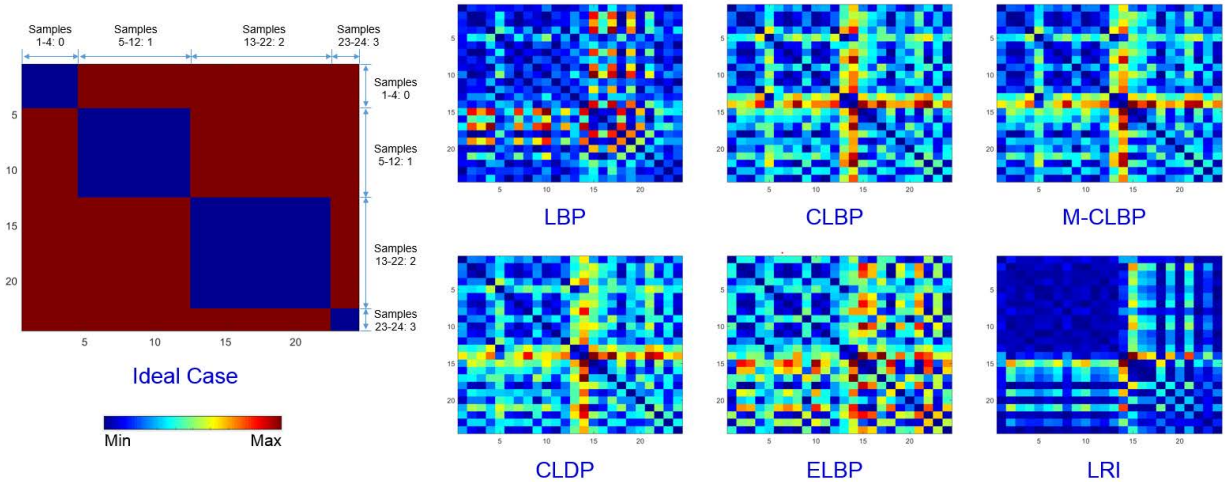
Binary Descriptors: Given that the six baseline features are not adequate to characterize the fabric samples, we then compared all samples using different binary descriptors. Local binary patterns uses binary strings to represent local patterns of intensity variations. Similar to LBP in concept, but instead of examining intensity variation, LRI measures distribution of local edges along various angles. We show the results of the ideal distance map and the corresponding actual map in Fig. 5.8a, Fig. 5.8b, and Fig. 5.8c regarding three



(a) Sample comparison using binary descriptor features regarding the "fiber length" property.



(b) Sample comparison using binary descriptor features regarding the "smoothness" property.



(c) Sample comparison using binary descriptor features regarding the "toweling effect" property.

Figure 5.8: Comparison of samples using binary descriptor features. In general, the local descriptors are inadequate to characterize the fabric samples.

surface texture attributes (i.e. fiber length, smoothness, and toweling), respectively. As we observe, compared to Tamura features, binary descriptor features exhibit more clues for differentiating between samples. As Fig. 5.8c shows, LRI roughly identifies samples from toweling level "0" and "1" but cannot differentiate between samples from these two levels. Hence, they are not promising for differentiating between the samples.

In summary, our exploration indicates that the six baseline features and binary descriptors are not adequate for texture image analysis. Hence, in next Sec. 5.4.3 we explore more advanced methods based on deep learning.

5.4.3 Experimental Results of Deep Learning-based Texture Representation

Implementation and Evaluation of Pretrained FV-CNN and SVM on CoMMonS Dataset

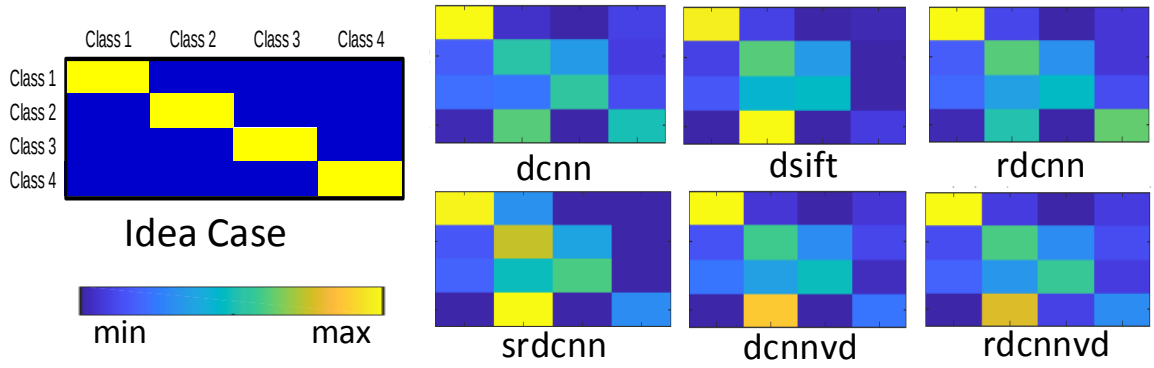
We studied pretrained CNN features including features from VGG-M architecture and features from VGG-VD architecture. As a byproduct, we also explored dense SIFT features, which are used in combination with the pretrained CNN features in some scenarios being investigated. The pretrained CNN features are among the very latest techniques for texture image analysis based on powerful deep learning and are proved highly effective for generic texture image analysis. In addition, their computation is efficient because of pretrained CNN models, avoiding the time-consuming training process typically required in deep learning. Similar to the conventional methods, we prepared software code that generates dense SIFT features and generic deep features from various pretrained CNN models, including VGG-M and VGG-VD.

We compared samples in deep feature space in comprehensive experiments: (1) we explored SVM classification using a variety of feature extraction algorithms combining pretrained CNN features and feature encoding/pooling; (2) we selected six different locations from each fabric sample for dataset preparation; and (3) we used six data split schemes to form training and testing datasets for classification experiments. We evaluated seven feature extraction algorithms combining pre-trained CNN features, feature encoding, and

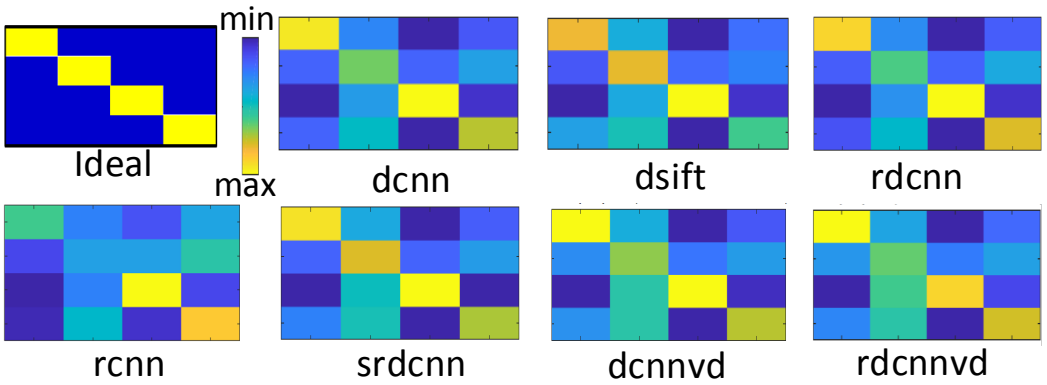
SVM as follows.

1. “rcnn”: a fully connected CNN (FC-CNN) from VGG_M without feature encoding;
2. “dcnn”: the combination of deep features from VGG_M and Fisher vector encoding;
3. “dsift”: the combination of SIFT features, PCA, and Fisher vector encoding;
4. “rdcnn”: the feature fusion of “rcnn” and “dcnn”;
5. “srdcnn”: the feature fusion of “rcnn”, “dcnn” and “dsift”.
6. “dcnnvd’: the combination of deep features from VGG_VD and Fisher vector encoding;
7. “rdcnnvd”: the feature fusion of a fully connected CNN (FC-CNN) from VGG_VD and “dcnnvd”.

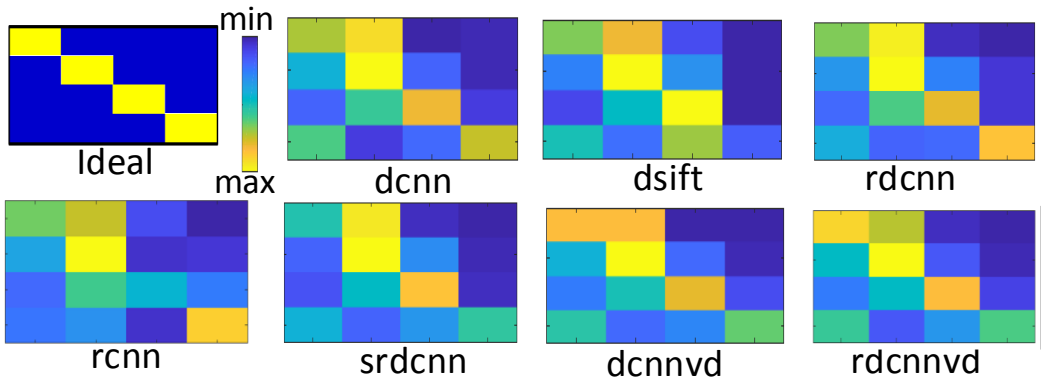
Confusion Matrices of Different FV-CNN Schemes on Patch Classification: A confusion matrix is a table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). Similar to the distance map mentioned in 5.4.2, the matrix makes it easy to see if the system is confusing two classes. We show confusion matrices of the ideal case and the corresponding actual case in Fig. 5.9a, Fig. 5.9b, and Fig. 5.9c regarding three surface texture properties (i.e. fiber length, smoothness, and toweling), respectively. In contrast to Tamura features (shown in Fig. 5.7a, Fig. 5.7b, and Fig. 5.7c) and binary descriptor features (shown in Fig. 5.8a, Fig. 5.8b, and Fig. 5.8c), it is obvious that FV-CNN features show more correlation between ideal confusion matrices and actual ones, especially presenting more highlighted areas in the diagonal of the matrix. Meanwhile, in the non-diagonal regions, various FV-CNN features show different degrees of error results. To provide an overall evaluation of classification performance, classification accuracies are calculated and compared next.



(a) Sample comparison using FV-CNN features regarding the "fiber length" property (zoom 50, rotation-30°).



(b) Sample comparison using FV-CNN features regarding the "smoothness" property (zoom 50, rotation-30°).

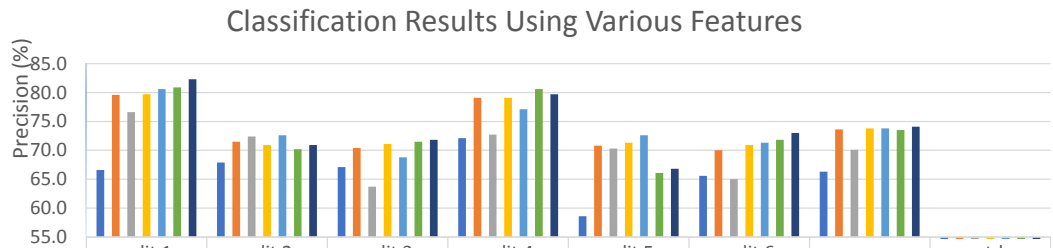


(c) Sample comparison using FV-CNN features regarding the "toweling effect" property (zoom 50, rotation-30°).

Figure 5.9: Confusion matrices using FV-CNN features.

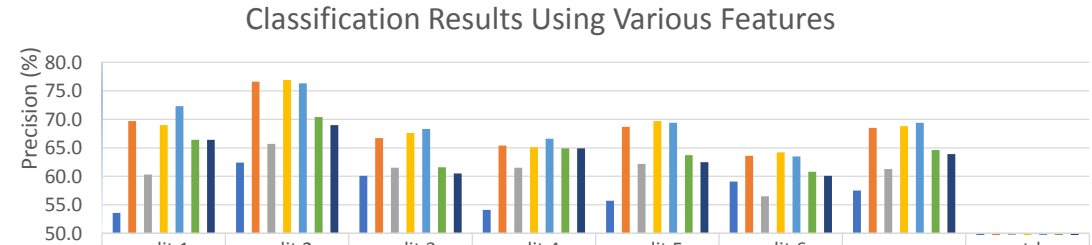
Overall Comparison of Different FV-CNN Schemes on Patch Classification:

We show the patch classification results including recognition accuracy and standard deviation noted "ave" and "std" for short under the condition of "zoom 50" and "rotation -30°" (denoted R-30) in Fig. 5.10a, Fig. 5.10b, and Fig. 5.10c. In Fig. 5.10a, image patches



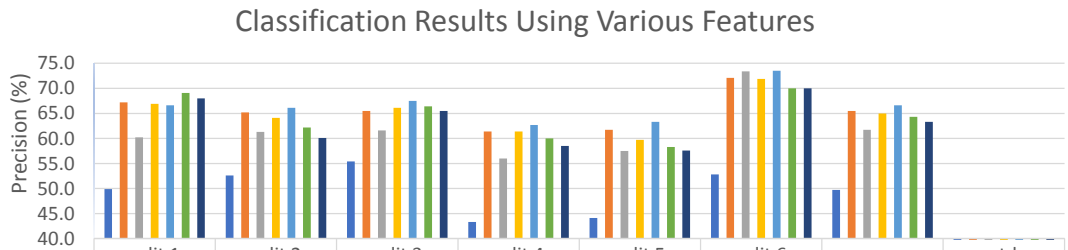
	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	66.6	67.9	67.1	72.1	58.6	65.6	66.3	4.0
dcnn	79.6	71.5	70.4	79.1	70.8	70.0	73.6	4.1
dsift	76.6	72.4	63.7	72.7	70.3	65.0	70.1	4.5
rdcnn	79.7	70.9	71.1	79.1	71.3	70.9	73.8	3.9
srdcnn	80.6	72.6	68.8	77.1	72.6	71.3	73.8	3.9
dcnnvd	80.9	70.2	71.5	80.6	66.1	71.8	73.5	5.4
rdcnnvd	82.3	70.9	71.8	79.7	66.8	73.0	74.1	5.3

(a) FV-CNN+SVM results on the "fiber length" property (zoom 50, rotation-30°)



	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	53.6	62.4	60.1	54.1	55.7	59.1	57.5	3.2
dcnn	69.7	76.6	66.7	65.4	68.7	63.6	68.5	4.2
dsift	60.3	65.7	61.5	61.5	62.2	56.5	61.3	2.7
rdcnn	69.0	76.9	67.6	65.1	69.7	64.2	68.8	4.1
srdcnn	72.3	76.3	68.3	66.6	69.4	63.5	69.4	4.1
dcnnvd	66.4	70.4	61.6	64.9	63.7	60.8	64.6	3.2
rdcnnvd	66.4	69.0	60.5	64.9	62.5	60.1	63.9	3.2

(b) FV-CNN+SVM results on the "smoothness" property (zoom 50, rotation-30°)



	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	49.9	52.6	55.4	43.3	44.1	52.8	49.7	4.5
dcnn	67.2	65.2	65.5	61.4	61.7	72.1	65.5	3.6
dsift	60.2	61.3	61.6	56.0	57.5	73.4	61.7	5.6
rdcnn	66.9	64.1	66.1	61.4	59.7	71.9	65.0	4.0
srdcnn	66.6	66.1	67.5	62.7	63.3	73.5	66.6	3.5
dcnnvd	69.1	62.2	66.4	60.0	58.3	70.0	64.3	4.5
rdcnnvd	68.0	60.1	65.5	58.5	57.6	70.0	63.3	4.8

(c) FV-CNN+SVM results on the "toweling" property (zoom 50, rotation-30°)

Figure 5.10: FV-CNN+SVM recognition results (zoom 50, rotation-30°).

were classified as of different fiber lengths and performance was averaged over all six split schemes. In Fig. 5.10b, image patches were classified as of different levels of smoothness. Similarly, in Fig. 5.10c, image patches were classified as of different levels of toweling effect. We have several observations as below:

1. In terms of different surface properties, the best feature extraction algorithm is different; the best feature extraction algorithm ("rdcnnvd") achieved an average precision of $74.1 \pm 5.3\%$ regarding "fiber length"; the best feature extraction algorithm ("srdcnn") achieved an average precision of $69.4 \pm 4.1\%$; and the best feature extraction algorithm ("srdcnn") achieved an average precision of $66.6 \pm 3.5\%$;

2. Different properties have different degrees of challenging recognition difficulties; for example, "fiber length" is the easiest property among the three to differentiate while "toweling effect" is the hardest under this setting;

3. Compared with "rcnn", the algorithm without feature encoding, other algorithms with feature encoding achieve higher recognition performance on the three surface properties; even handcrafted features like "dsift" involving feature encoding performs better than "rcnn";

4. Deep learning features with more layers not always perform better since "rdcnnvd" only achieves the best on "fiber length" and "srdcnn" achieves the best on both "smoothness" and "toweling";

5. The feature fusion of "dsift" and FV-CNN features bring more discriminative ability of surface properties. For example, compared to "rdcnn", "srdcnn" brings 0.6% and 1.6% improvement on recognizing"smoothness" and "toweling effect", respectively.

6. The recognition standard deviations for "smoothness" are always correspondingly smaller than those of the other two properties, which indicates that locations reflected

by different splits affect less on recognizing "smoothness" because of its relative uniformity on the surface.

To conduct a comprehensive evaluation of different settings, we also test on "Zoom 200, R-30" and "Zoom 50, R+60". We show the patch classification results including recognition accuracy and standard deviation noted "ave" and "std" for short under the condition of "zoom 50" and "rotation +60°" (denoted R+60) in Fig. 5.11a, Fig. 5.11b, and Fig. 5.11c. We have several observations as below:

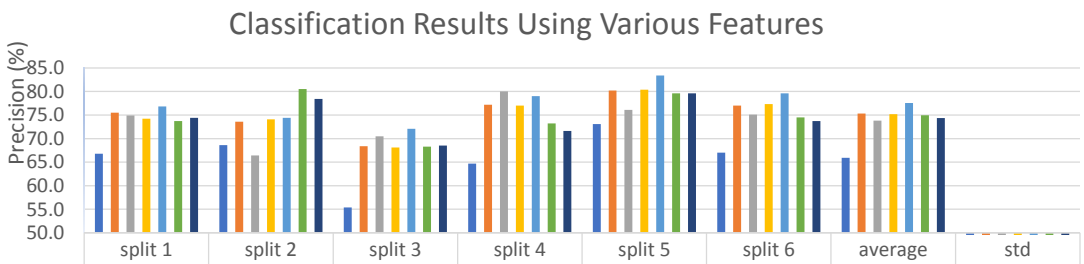
1. Similar to setting "Zoom 200, R-30", the best feature extraction algorithm is different regarding different surface properties; in terms of "fiber length", the best feature extraction algorithm ("rdcnnvd") achieved an precision of $61.0 \pm 5.3\%$; the best feature extraction algorithm ("rdcnn") achieved an average precision of $52.5 \pm 6.1\%$; and the best feature extraction algorithm ("dcnnvd") achieved an average precision of $53.1 \pm 3.0\%$;

2. Similar to setting "Zoom 50, R-30" and "Zoom 50, R+60", "fiber length" is the easiest property among the three to differentiate while "toweling effect" is the hardest under this setting;

3. Similar to setting "Zoom 50, R-30" and "Zoom 50, R+60", other algorithms with feature encoding achieve higher recognition accuracies over "rcnn" on the three surface properties; even handcrafted features like "dsift" involving feature encoding performs better than "rcnn", which supports the power of feature encoding;

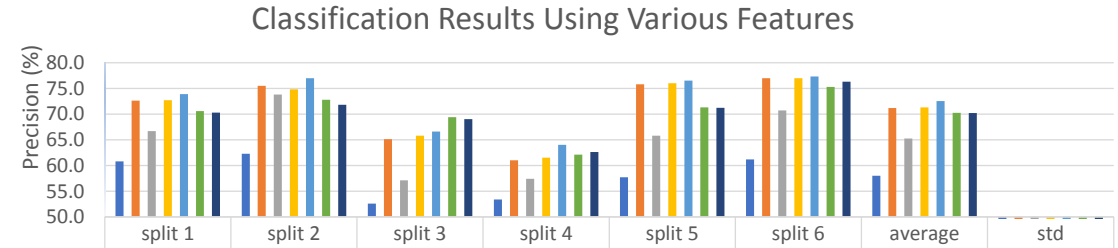
4. Deep learning features with more layers sometimes perform better such as "rdcnnvd" achieves the best on "fiber length" and "dcnnvd" achieves the highest on "toweling effect";

5. Different from the case of zooming level 50, the feature fusion of "dsift" and FV-CNN features not always bring more discriminative ability of surface properties. For



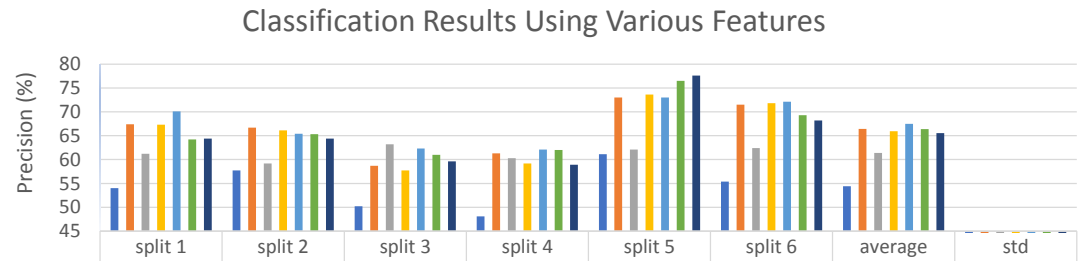
	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	66.8	68.6	55.4	64.7	73.1	67.0	65.9	5.0
dcnn	75.5	73.6	68.4	77.2	80.2	77.0	75.3	3.4
dsift	74.9	66.4	70.5	80.0	76.1	75.1	73.8	4.0
rdcnn	74.2	74.1	68.1	77.0	80.4	77.3	75.2	3.5
srdcnn	76.8	74.4	72.1	79.0	83.4	79.6	77.6	3.4
dcnnvd	73.7	80.5	68.3	73.2	79.6	74.5	75.0	3.8
rdcnnvd	74.4	78.4	68.5	71.6	79.6	73.7	74.4	3.5

(a) FV-CNN+SVM results on the "fiber length" property (zoom 50, rotation+60°)



	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	60.8	62.3	52.6	53.4	57.7	61.2	58.0	3.8
dcnn	72.6	75.5	65.1	61.0	75.8	77.0	71.2	6.0
dsift	66.7	73.8	57.1	57.4	65.8	70.7	65.3	6.2
rdcnn	72.7	74.8	65.8	61.5	76.0	77.0	71.3	5.7
srdcnn	73.9	77.0	66.6	64.0	76.5	77.3	72.6	5.3
dcnnvd	70.6	72.8	69.4	62.1	71.3	75.3	70.3	4.1
rdcnnvd	70.3	71.8	69.0	62.6	71.2	76.3	70.2	4.1

(b) FV-CNN+SVM results on the "smoothness" property (zoom 50, rotation+60°)



	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	54.0	57.7	50.2	48.1	61.1	55.4	54.4	4.4
dcnn	67.4	66.7	58.7	61.3	73.0	71.5	66.4	5.1
dsift	61.2	59.2	63.2	60.3	62.1	62.4	61.4	1.3
rdcnn	67.3	66.1	57.7	59.2	73.6	71.8	66.0	5.9
srdcnn	70.1	65.4	62.3	62.1	73.0	72.1	67.5	4.4
dcnnvd	64.2	65.3	61.0	62.0	76.5	69.3	66.4	5.2
rdcnnvd	64.4	64.4	59.6	58.9	77.6	68.2	65.5	6.3

(c) FV-CNN+SVM results on the "toweling" property (zoom 50, rotation+60°)

Figure 5.11: FV-CNN+SVM recognition results (zoom 50, rotation +60°).

example, when recognizing "smoothness", "srdcnn" (51.7%) performs 0.8% worse than "rdcnn" (52.5%) due to fused "dsift" features.

6. Under the same rotation angle but different zooming levels, recognition accuracies of "zoom 200" are much lower (more than 13%) than those of "zoom 50". This indicates that "zoom 50" already provides sufficient details for differentiating surface properties while "zoom 200" focuses too small local regions and leads information loss.

We show the patch classification results including recognition accuracy and standard deviation noted "ave" and "std" for short under the condition of "zoom 200" and "rotation -30°" (denoted R-30) in Fig. 5.12a, Fig. 5.12b, and Fig. 5.12c. We have several observations as below:

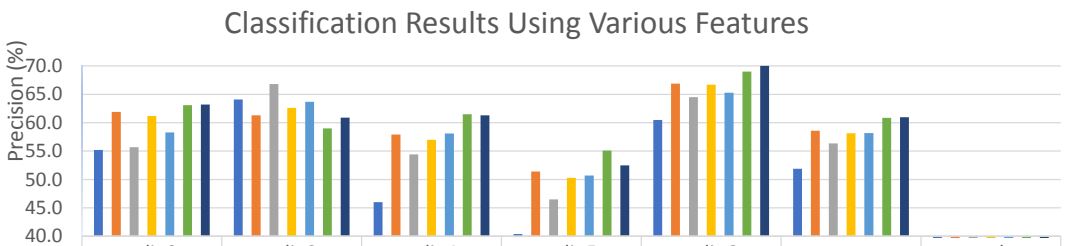
1. In terms of different surface properties, the best feature extraction recognition accuracies $77.6 \pm 3.4\%$, $72.6 \pm 5.3\%$, and $67.5 \pm 4.4\%$ are all achieved by algorithm ("srdcnnvd");

2. Similar to setting "Zoom 50, R-30", "fiber length" is the easiest property among the three to differentiate while "toweling effect" is the hardest under this setting;

3. Similar to setting "Zoom 50, R-30", other algorithms with feature encoding achieve higher recognition accuracies over "rcnn" on the three surface properties; even hand-crafted features like "dsift" involving feature encoding performs better than "rcnn";

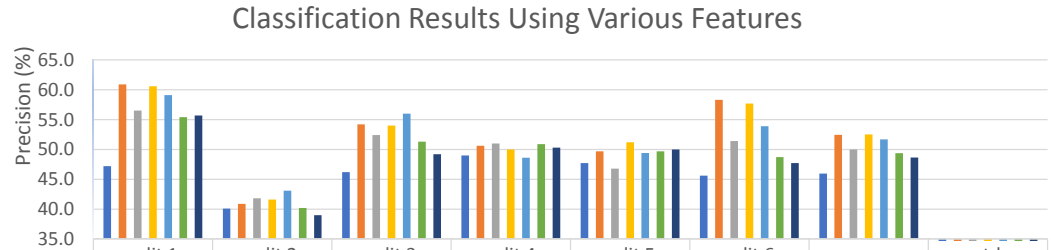
4. Deep learning features with more layers do not always perform better since "srdcnn" achieves the best on the three properties;

5. The feature fusion of "dsift" and FV-CNN features bring more discriminative ability of surface properties. For example, compared to "rdcnn", "srdcnn" brings 2.4%, 1.3%, and 1.5% improvement on recognizing "fiber length", "smoothness", and "toweling effect", respectively.



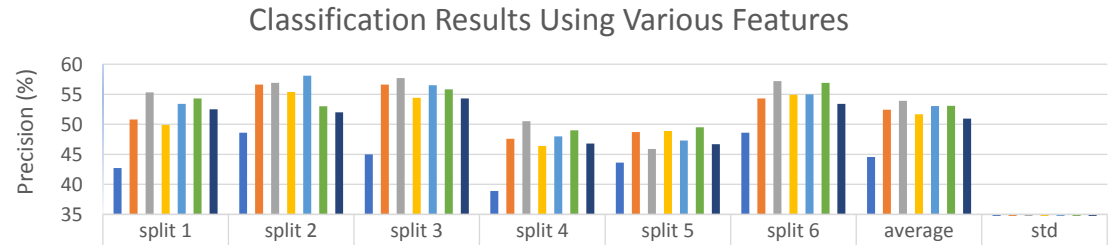
	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	55.2	64.1	46.0	40.4	60.5	51.9	8.6
dcnn	61.9	61.3	57.9	51.4	66.9	58.6	5.5
dsift	55.7	66.8	54.4	46.5	64.5	56.4	7.2
rdcnn	61.2	62.6	57.0	50.3	66.7	58.1	6.0
srdcnn	58.3	63.7	58.1	50.7	65.3	58.2	5.2
dcnnvd	63.1	59.0	61.5	55.1	69.0	60.9	4.5
rdcnnvd	63.2	60.9	61.3	52.5	70.0	61.0	5.3

(a) FV-CNN+SVM results on "fiber length" (zoom 200, rotation-30°)



	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	47.2	40.1	46.2	49.0	47.7	45.6	46.0	2.8
dcnn	60.9	40.9	54.2	50.6	49.7	58.3	52.4	6.5
dsift	56.5	41.8	52.4	51.0	46.8	51.4	50.0	4.6
rdcnn	60.6	41.6	54.0	50.0	51.2	57.7	52.5	6.1
srdcnn	59.1	43.1	56.0	48.6	49.4	53.9	51.7	5.3
dcnnvd	55.4	40.2	51.3	50.9	49.7	48.7	49.4	4.6
rdcnnvd	55.7	39.0	49.2	50.3	50.0	47.7	48.7	5.0

(b) FV-CNN+SVM results on "smoothness" (zoom 200, rotation-30°)



	split 1	split 2	split 3	split 4	split 5	split 6	average	std
rcnn	42.7	48.6	45.0	38.9	43.6	48.6	44.6	3.4
dcnn	50.8	56.6	56.6	47.6	48.7	54.3	52.4	3.6
dsift	55.3	56.9	57.7	50.5	45.9	57.2	53.9	4.3
rdcnn	49.9	55.4	54.4	46.4	48.9	54.9	51.7	3.4
srdcnn	53.4	58.1	56.5	48.0	47.3	55.0	53.1	4.1
dcnnvd	54.3	53.0	55.8	49.0	49.5	56.9	53.1	3.0
rdcnnvd	52.5	52.0	54.3	46.8	46.7	53.4	51.0	3.1

(c) FV-CNN+SVM results on "toweling" (zoom 200, rotation-30°)

Figure 5.12: FV-CNN+SVM recognition results (zoom 200, rotation-30°).

6. Under the same zooming level but different rotation angles, the magnitude of recognition accuracies are of the same level, which supports the rotation invariance of FV-CNN features.

In summary, pre-trained deep features, in combination with feature encoding techniques, are capable of capturing the unique characteristics of fabrics in terms of fabric length, smoothness, and toweling effect with a reasonable accuracy.

From Patch Classification to Sample Characterization:

In the classification experiments, patches are classified as of certain fiber length (or of certain level of smoothness / toweling effect). As illustrated in Fig. 5.13, combining classification results for all patches extracted from one fabric sample through decision fusion, we are able to label (or characterize) the fabric sample as of certain fiber length (or of certain level of smoothness / toweling effect). We performed a preliminary exploration on this sample characterization in terms of smoothness by using one data split scheme and one feature extraction algorithm together with an efficient majority voting as decision fusion. Yet, the result is encouraging as shown in Fig. 5.14, which lists both human ratings and computer rating w.r.t each sample. Most samples were automatically and correctly labels while "S6", "S14","S15", and "S23" were not. 20 of the 24 samples were labeled correctly, which is 83.3% in accuracy.

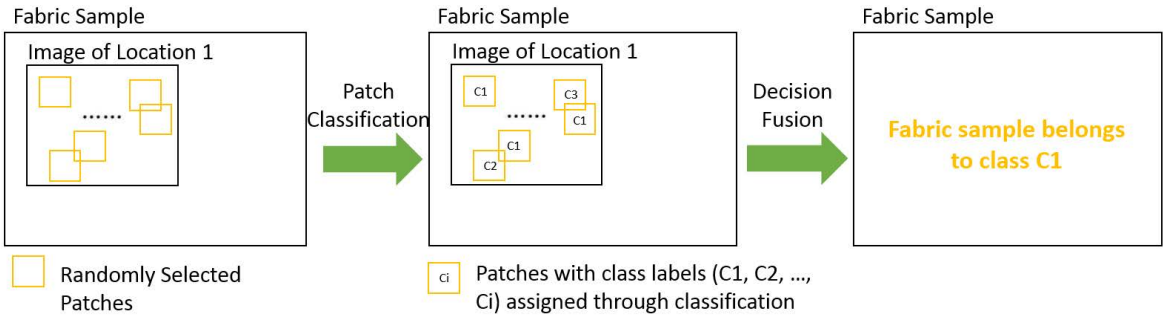


Figure 5.13: An illustration of sample characterization method

Sample	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
Human Rating	2	1	1	1	1	4	2	2	2	2	2	2
Computer Rating	2	1	1	1	1	1	2	2	2	2	2	2

Sample	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
Human Rating	2	3	4	1	2	3	2	4	1	2	3	2
Computer Rating	2	4	2	1	2	3	2	4	1	2	2	2

Figure 5.14: An illustration of sample characterization result

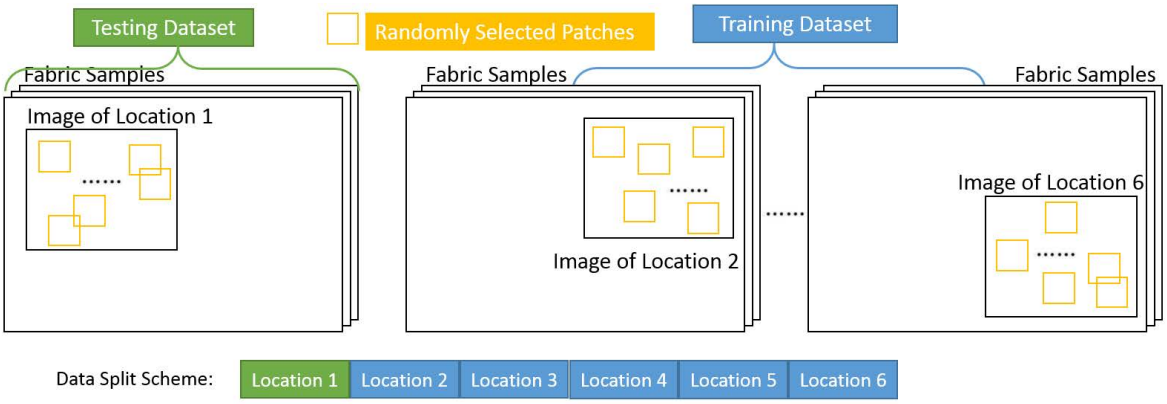
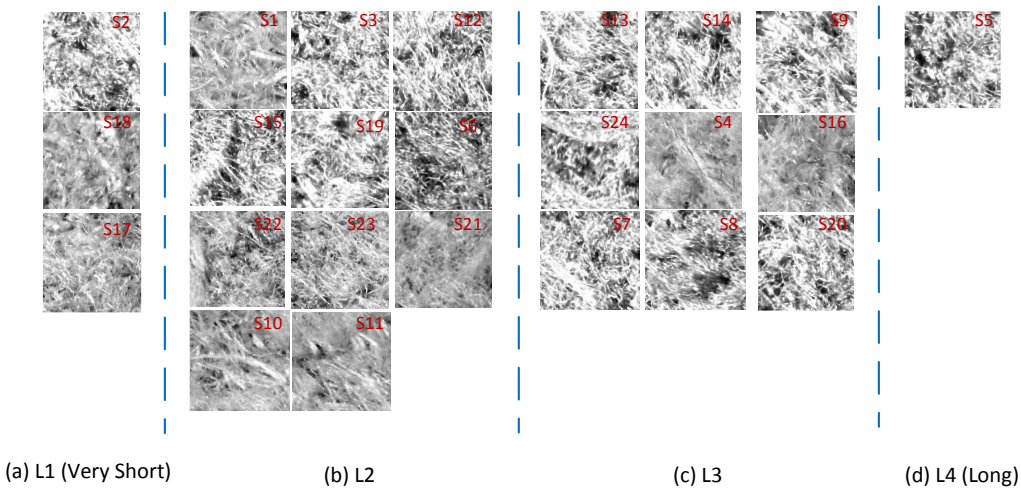


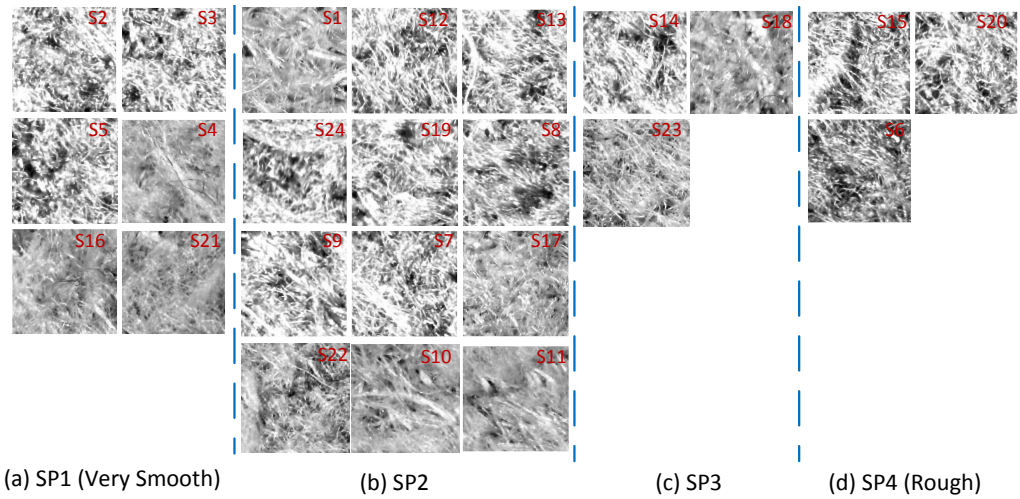
Figure 5.15: An illustration of six-fold validation.

Implementation and Evaluation of End-to-end Deep Learning Methods on CoMMonS dataset

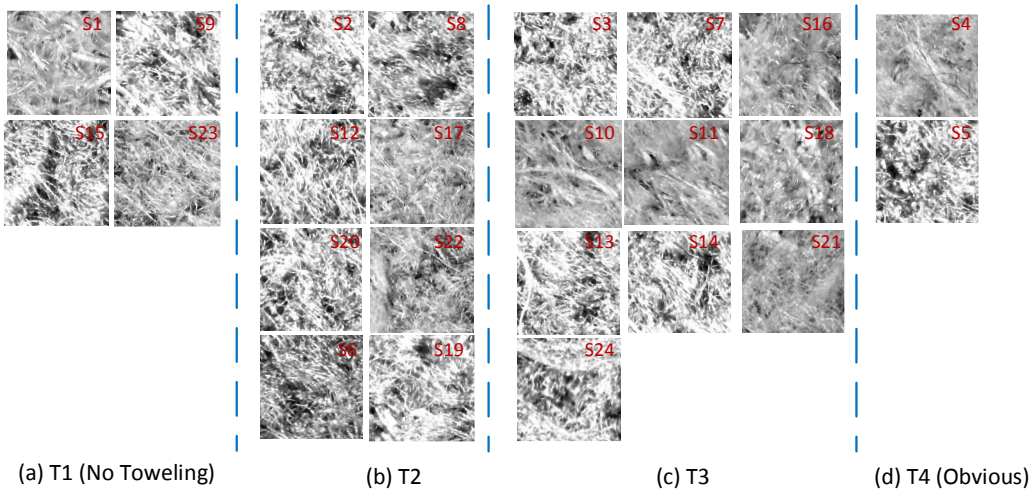
Data Preparation: We performed six-fold cross validation for training and testing splits shown in Fig. 5.15. In our work, images were acquired at six non-overlapping locations of each fabric sample. Taking one fold as an example, patches of size 300×300 extracted from images of location 2, location 3, $\cdots$, and location 6 are used to form the training set, which is used to train and learn the feature encoder and the classifier. Then, patches of the same size extracted from images of location 1 of all fabric samples are used to form the testing set, which is used to test how good the feature encoder and the classifier are for predicting the characteristics of a given patch. Patch examples of different "smoothness" levels from "very smooth" to "rough" are shown in Fig. 5.16b. Similarly, patch examples of different "fiber length" and "toweling effect" properties are shown in Fig. 5.16a and Fig. 5.16c. We repeat this procedure for other five data splits. Such six-fold cross validation is used for



(a) Patch examples regarding the "fiber length" property.



(b) Patch examples regarding the "smoothness" property.



(c) Patch examples regarding the "toweling effect" property.

Figure 5.16: Texture patch examples regarding texture properties.

texture classification evaluation of different representation algorithms later. The training set is used to train FV-CNN or end-to-end learning approaches and the testing dataset is used to test how good they are for predicting the characteristics of a given patch.

Performance Evaluation: We run experiments on a PC (Nvidia GeForce GTX1070, RAM: 8GB) and evaluate state-of-the-art methods including FV-CNN [108] and DEP [16]. The experimental settings for DEP [16] and MuLTER are: learning rate starting at 0.1 and decaying every 10 epochs (step = 10) by a factor of 0.1, batch size 16, and the total number of epochs 30. The number of codewords K is set to 32. For a fair comparison to our MuLTER method using fully connected layer and softmax as a classifier, we extract FV-CNN [108] features combining pre-trained CNN features and feature encoding and apply the same classifier and the same setting (i.e. staring learning rate = 0.1, step = 10, epochs = 30, and batch size =16). Regarding FV-CNN, we evaluate two pretrained models, VGG-M and VGG-VD. The extensive comparisons of the four methods for each surface property and two zooming levels are shown in Table 5.5 to Table 5.10, respectively.

Table 5.5: Comparison with state-of-the-art algorithms in $\%$ on the CoMMonS dataset for the "fiber length" property. (zoom: 50, rotation: -30°)

Data Splits	FV-CNN [108] (VGG-M)	FV-CNN [108] (VGG-VD)	DEP [16]	**MuLTER** [115]
Split 1	70.5	67.4	58.4	65.3
Split 2	52.4	58.9	58.2	58.7
Split 3	64.2	55.3	50.3	52.1
Split 4	59.2	55.8	67.6	72.4
Split 5	58.2	59.2	63.2	66.6
Split 6	63.4	54.2	44.2	57.1
Average	61.3±5.6	58.5±4.4	57.0±7.8	**62.0±6.7**

Taking the comparison regarding the "Smoothness" property under the zooming level "50" as an example, we discuss the performance of different methods on fabric surface property characterization. The classification accuracy of all six splits and their average are shown in Table 5.6. The best feature extraction algorithm (MuLTER) achieved an average accuracy of 59.0%, which is 2.1% higher over its closest counterpart, DEP [16],

Table 5.6: Comparison with state-of-the-art algorithms in % on the CoMMonS dataset for the "smoothness" property. (zoom: 50, rotation: -30°)

Data Splits	FV-CNN [108] (VGG-M)	FV-CNN [108] (VGG-VD)	DEP [16]	**MuLTER** [115]
Split 1	52.5	44.8	54.8	55.6
Split 2	55.7	60.8	57.5	65.6
Split 3	59.4	56.0	57.3	60.8
Split 4	56.9	53.3	51.9	54.4
Split 5	55.6	52.9	54.2	58.1
Split 6	56.3	50.2	65.6	59.6
Average	56.1±2.0	53.0±4.9	56.9±4.3	**59.0±3.7**

Table 5.7: Comparison with state-of-the-art algorithms on the CoMMonS dataset regarding the "toweling" property. (zoom: 50, rotation: -30°)

Data Splits	FV-CNN [108] (VGG-M)	FV-CNN [108] (VGG-VD)	DEP [16]	**MuLTER** [115]
Split 1	58.4	51.6	51.6	61.6
Split 2	57.8	53.4	55.3	57.5
Split 3	55.9	51.9	47.5	55.6
Split 4	48.1	57.2	41.9	48.8
Split 5	56.3	51.0	52.2	56.3
Split 6	60.0	62.8	60.0	57.8
Average	56.1±3.8	54.7±4.2	51.4±5.7	**56.3±3.8**

Table 5.8: Comparison with state-of-the-art algorithms in % on the CoMMonS dataset for the "fiber length" property. (zoom: 200, rotation: -30°)

Data Splits	FV-CNN [108] (VGG-M)	FV-CNN [108] (VGG-VD)	DEP [16]	**MuLTER** [115]
Split 1	42.1	45.5	41.8	46.3
Split 2	46.1	46.3	53.2	51.8
Split 3	53.7	52.6	57.7	58.2
Split 4	44.2	52.9	48.9	63.9
Split 5	43.2	51.3	47.6	52.4
Split 6	53.4	60.0	51.3	55.0
Average	47.1±4.7	51.4±4.8	50.1±4.9	**54.6±5.5**

2.9% better than FV-CNN(VGG-M) [108], and 6.0% more accurate than FV-CNN(VGG-VD) [108]. From Table 5.5 to Table 5.10, we highlight the highest accuracy in bold. Our method, MuLTER, achieved the best in all six tables, while the other methods in comparison did not perform as consistently. The improvement over the second best method is as

Table 5.9: Comparison with state-of-the-art algorithms in % on the CoMMonS dataset for the "smoothness" property. (zoom: 200, rotation: -30°)

Data Splits	FV-CNN [108] (VGG-M)	FV-CNN [108] (VGG-VD)	DEP [16]	**MuLTER** [115]
Split 1	50.4	47.3	51.9	51.5
Split 2	42.5	40.0	46.5	46.0
Split 3	50.8	44.0	56.9	52.3
Split 4	46.0	50.2	51.5	55.2
Split 5	41.7	45.6	48.8	49.0
Split 6	51.3	43.3	48.8	52.9
Average	47.1± 4.0	45.1±3.2	50.7±3.3	**51.2±2.9**

Table 5.10: Comparison with state-of-the-art algorithms on the CoMMonS dataset for the "toweling" property. (zoom: 200, rotation: -30°)

Data Splits	FV-CNN [108] (VGG-M)	FV-CNN [108] (VGG-VD)	DEP [16]	**MuLTER** [115]
Split 1	39.1	47.8	39.4	47.5
Split 2	50.6	45.6	41.9	47.5
Split 3	50.3	50.9	40.9	42.8
Split 4	37.5	38.4	49.4	47.2
Split 5	38.8	43.8	45.9	44.7
Split 6	49.7	46.9	53.1	53.8
Average	44.3±5.9	45.6±3.9	45.1±4.9	**47.3±3.4**

follows: 0.7% in Table 5.5, 2.1% in Table 5.6, 0.2% in Table 5.7, 3.2% in Table 5.8, 0.5% in Table 5.9, and 1.7% in Table 5.10. This supports our claim that the end-to-end texture representation with multi-level feature fusion is capable of capturing the unique characteristics of fabrics in terms of fabric length, smoothness, and toweling effect better than its state-of-the-art counterparts. Although the overall accuracy rates are not high across the experiments, they are reasonable considering the challenging nature of the dataset. Interestingly, for all experiments, the performance is typically higher for fiber length, followed by smoothness, and lowest for toweling effect. We believe this is because fiber length is the most well-defined and most apparent among the three properties. On the contrary, toweling effect is an irregularity condition of the fabric, which is typically distributed very sparsely among the data samples. Additionally, comparing the classification accuracy of the same property under different zooming levels, we observe that patches with zooming level "200"

are more difficult to be differentiated than those of zooming level "50". We believe this performance degradation is caused by a lack of necessary global or macro information. Therefore, capturing fine details with large zooming levels and maintaining global/macro information is a trade-off that needs to be carefully considered when designing the data collection protocol.

5.5 Summary

In this chapter, we formulated the problem of characterizing a material in terms of a certain property as a fine-grained texture classification problem. Intrinsically, fine-grained texture classification is a challenging problem because of small inter-class appearance variations between subcategories and latent material properties. In our work, we studied material characterization in the context of an automated fabric surface characterization system. Using microscopy imaging, we created a dataset of fabric surfaces, CoMMonS, a first-of-its-kind public dataset geared towards this understudied problem. We assessed the state-of-the-art deep learning-based texture representation techniques using CoMMonS, and demonstrated that they are inadequate for such characterization tasks. In addition, we designed an innovative deep learning network architecture, MuLTER, that extracts both low-level and high-level CNN features to achieve a multi-level texture representation. With MuLTER, we were able to achieve enhanced performance over the state-of-the-art deep learning algorithms not only for material characterization using CoMMonS, but also for more general texture/material recognition. This proved the value of integrating CNN features from multiple levels with the embedment of a learnable encoding module at each level for texture representation. Our exploration here provided a unique benchmark for material characterization and fine-grained texture classification, which can be useful for a wide rage of real-world application scenarios.

CHAPTER 6
CONCLUSION

In Chapter 3, we investigated the role of binary representation in texture tracking and classification. The performance of texture tracking depends heavily on the binary representation of interest points, which are commonly identified by a detector with specific criterion. The binary representation of interest points is required to have sufficient discriminative power, which determines the matching accuracy of corresponding interest points in two adjacent frames. In addition to texture tracking, the discriminative power of binary representation is also crucial for texture classification. Binary representation algorithms (e.g., LBP variants) typically extract texture patterns in binary from densely sampled image patches over an image and utilize the corresponding distribution to generate full image representation. Our contributions on binary representation and its applications in texture tracking and classification are summarized as follows:

1. To achieve motion tracking of textures, we introduced an efficient, robust, and accurate feature-based approach to track texture patterns and provided associated motion information in terms of position and orientation. Our contributions include the following aspects: (1) an innovative framework that integrates a lattice detection module to accomplish texture tracking in a texture pattern-based coordinate system instead of a pixel-based system, to ensure robustness to local texture deformation; (2) a novel algorithm for fast and efficient lattice detection, achieved by constraining on local appearance similarities and global topology; and (3) an extensive comparative study evaluating various methods of interest point detection and description for their applicability to the texture tracking problem, demonstrating its high potential for real-time visual tracking problems. During the process of texture tracking, given an appropriate interest point detector, the selection of interest point descriptor is an important

step for accurate interest point matching and highly computational efficiency. Binary representation (e.g. BRISK) used in this chapter is such a good option here because of its fast feature extraction and distance calculation. Therefore, for the problem of real-time texture tracking, binary representation performs well on describing local textures in terms of discriminative ability and computational efficiency.

2. To better involve multi-scale information or handle scale variations in the task of texture classification, we developed two binary representation approaches. Different from popular LBP variants like CLBP encoding local binary patterns in each scale separately, we designed a texture descriptor called CLDP to involve multi-scale textural information in local regions. CLDP introduces a new component, the directional derivative pattern to extract the patterns of two neighboring scales in the same direction to calculate the corresponding directional cross-scale correlation, which characterizes local texture smoothness along each direction. By including the new component, CLDP covers four types of binary representations: the sign, magnitude, and local directional derivative of local intensity differences and the intensity values of center pixels, respectively. For a local region around one pixel, the information from these four binary representations is fused though a joint encoding of binary codes and such encoding encodes the combined information into a feature value representing the pattern of this local region. When compared with state-of-the-art uni- and multi-scale texture descriptors on the Outex database [87], CLDP outperforms its uni-scale counterparts like CLBP in terms of texture classification accuracy without adding computational complexity and it is comparable to the multi-scale texture descriptors on the classification accuracy, which supports our claim that binary representation involving multi-scale information improves its discriminative ability of textural information.

3. Since commonly used binary representation like ELBP keeps the settings of feature

extraction unchanged for all images, scale variations between images may degenerate the classification performance of binary representation. Therefore, we developed a binary representation approach named scale-selective extended local binary patterns (SSELBP) to generate an image representation robust to scale variations. SSELBP builds a scale space through applying Gaussian filters and extracts binary patterns at each scale. Similar to CLDP, the representation for each scale utilize binarized local intensity differences and involves multi-scale information. For the purpose of scale invariance, a key step of SSELBP is a maximum pooling strategy on the features across all scales. This step obtains scale-invariant features. With the experimental results on two texture databases with scale variations, we observed that compared to state-of-the-art descriptors, SSELBP achieved comparable accuracy with much lower-dimensional features. Therefore, binary representation plays an important role in discriminative, efficient, and robust representation for texture classification.

4. To achieve texture tracking for the purpose of automatic motion surveillance, we introduced an efficient, robust, and accurate feature-based approach to track individual fabric threads and provided associated motion information in terms of position and orientation. Our contributions include the following aspects: (1) an innovative framework that integrates a lattice detection module to accomplish fabric tracking in a thread-based coordinate system instead of a pixel-based system, to ensure robustness to local fabric deformation; (2) a novel algorithm for fast and efficient lattice detection for thread counting, achieved by constraining on local appearance similarities and global topology; and (3) an extensive comparative study evaluating various methods of interest point detection and description for their applicability to the fabric tracking problem, demonstrating its high potential for automatic real-time textile manufacturing. During the process of texture tracking, given an appropriate interest point detector, the selection of interest point descriptor is an important step for accurate interest point matching and highly computational efficiency. Binary rep-

resentation such as BRISK used in this chapter is such a wise option here because of its fast feature extraction and distance calculation. Therefore, for the problem of real-time texture tracking, binary representation performs well on describing local textures in terms of dicsriminative ability and computational efficiency.

In Chapter 4, we introduce a novel local texture representation method, block intensity and gradient difference (BIGD), which achieves great distinctiveness and computational efficiency. Compared with other algorithms mentioned above, our main contribution is efficiently captures un-quantized gradient difference features in BIGD. The gradient difference captures the variations of gradients in a local patch and improves distinctiveness. Descriptors such as histogram of orientated gradients (HOG) and SIFT utilize gradient-based features to capture the orientation information. However, quantized orientations in them result in information loss. Our BIGD method extracts intensity- and gradient-difference features at multi-orientations without quantization and retains the discriminative power of features. We have three major steps to evaluate the performance of BIGD on texture classification. First, we randomly select block pairs within an image patch. For each block, we extract a five-dimensional feature vector that contains the means of intensity, gradient, and absolute gradient values. Since we randomly sample pairwise blocks with multiple scales and orientations, the BIGD descriptor that concatenates the differences of feature vectors extracted from block pairs reveals underlying texture structures at different spatial granularities and orientations. Second, we encode the BIGD descriptors of local patches into an entire image representation using feature encoding modules such as vectors of locally aggregated descriptors (VLAD) [55] or improved Fisher vectors (IFV) [94][95]. Third, the texture images are classified using a linear support vector machine (SVM). The superior performance of our approach was demonstrated by an extensive evaluation on public texture databases. In future work, we will improve the discriminative power of the BIGD descriptor on rotation variations and extend it to other computer vision tasks such as object recognition.

In Chapter 5, we formulated the problem of characterizing a material in terms of a certain property as a fine-grained texture classification problem. Intrinsically, fine-grained texture classification is a challenging problem because of small inter-class appearance variations between subcategories and latent material properties. In our work, we studied material characterization in the context of an automated fabric surface characterization system. Using microscopy imaging, we created a dataset of fabric surfaces, CoMMonS, a first-of-its-kind public dataset geared towards this understudied problem. We assessed the state-of-the-art deep learning-based texture representation techniques using CoMMonS, and demonstrated that they are inadequate for such characterization tasks. In addition, we designed an innovative deep learning network architecture, MuLTER, that extracts both low-level and high-level CNN features to achieve a multi-level texture representation. With MuLTER, we were able to achieve enhanced performance over the state-of-the-art deep learning algorithms not only for material characterization using CoMMonS, but also for more general texture/material recognition. This proved the value of integrating CNN features from multiple levels with the embedment of a learnable encoding module at each level for texture representation. Our exploration here provided a unique benchmark for material characterization and fine-grained texture classification, which can be useful for a wide rage of real-world application scenarios.

REFERENCES

[1] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, 2009.

[2] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[3] H. Liu and F. Sun, *Robotic Tactile Perception and Understanding: A Sparse Coding Method*. 2018.

[4] D. Hu, L. Bo, and X. Ren, "Toward robust material recognition for everyday objects.," in *British Machine Vision Conference*, vol. 2, 2011, p. 6.

[5] X. Xie and M. Mirmehdi, "Texture exemplars for defect detection on random textures," in *International Conference on Pattern Recognition and Image Analysis*, 2005, pp. 404–413.

[6] C. Kampouris, S. Zafeiriou, A. Ghosh, and S. Malassiotis, "Fine-grained material classification using micro-geometry and reflectance," in *European Conference on Computer Vision*, 2016, pp. 778–792.

[7] W. Sun, B. Yao, B. Chen, Y. He, X. Cao, T. Zhou, and H. Liu, "Noncontact surface roughness estimation using 2d complex wavelet enhanced resnet for intelligent evaluation of milled metal surface quality," *Applied Sciences*, vol. 8, no. 3, p. 381, 2018.

[8] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi, "Deep filter banks for texture recognition, description, and segmentation," *International Journal of Computer Vision*, vol. 118, no. 1, pp. 65–94, 2016.

[9] L. Nanni, A. Lumini, and S. Brahnam, "Local binary patterns variants as texture descriptors for medical image analysis," *Artificial Intelligence in Medicine*, vol. 49, no. 2, pp. 117–125, 2010.

[10] G. AlRegib, M. Deriche, Z. Long, H. Di, Z. Wang, Y. Alaudah, M. A. Shafiq, and M. Alfarraj, "Subsurface structure analysis using computational interpretation and learning: A visual signal processing perspective," *IEEE Signal Processing Magazine*, vol. 35, no. 2, pp. 82–98, 2018.

[11] W. J. Book, R. C. Winck, M. D. Killpack, J. D. Huggins, S. L. Dickerson, S. Jayaraman, T. R. Collins, and R. J. Prado, "Automated garment manufacturing system using novel sensing and actuation," in *International Symposium on Flexible Automation*, 2010.

[12] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: A review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[13] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018.

[14] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3828–3836.

[15] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3606–3613.

[16] J. Xue, H. Zhang, and K. Dana, "Deep texture manifold for ground terrain recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 558–567.

[17] J. M. Dischler and F. Zara, "Real-time structured texture synthesis and editing using image-mesh analogies," *The Visual Computer*, vol. 22, no. 9-11, pp. 926–935, 2006.

[18] F. Solera, S. Calderara, and R. Cucchiara, "Structured learning for detection of social groups in crowd," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2013, pp. 7–12.

[19] C. He, S. Li, Z. Liao, and M. Liao, "Texture classification of polsar data based on sparse coding of wavelet polarization textons," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 8, pp. 4576–4590, 2013.

[20] U. Kandaswamy, D. A. A. A, and M. C. Lee, "Efficient texture analysis of sar imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 9, pp. 2075–2083, 2005.

[21] C. Chen, *Handbook of pattern recognition and computer vision*. 2015, pp. 235–276.

[22] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[23] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[24] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 1, pp. 25–39, 1983.

[25] Y. Xu, H. Ji, and C. Fermüller, "Viewpoint invariant texture description using fractal analysis," *International Journal of Computer Vision*, vol. 83, no. 1, pp. 85–100, 2009.

[26] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[28] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *International Journal of Computer Vision*, vol. 62, no. 1-2, pp. 61–81, 2005.

[29] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao, "WLD: A robust local image descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1705–1720, 2010.

[30] L. Liu, P. Fieguth, D. Hu, Y. Wei, and G. Kuang, "Fusing sorted random projections for robust texture and material classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 482–496, 2015.

[31] R. Mehta and K. Egiazarian, "Rotation invariant texture description using symmetric dense microblock difference," *IEEE Signal Processing Letters*, vol. 23, no. 6, pp. 833–837, 2016.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[33] H. Zhang, J. Xue, and K. Dana, "Deep ten: Texture encoding network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2896–2905.

[34] F. Perronnin and C.Dance, "Fisher kernels on visual vocabularies for image categorization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[35] J. Xue, H. Zhang, K. Dana, and K. Nishino, "Differential angular imaging for material recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 764–773.

[36] L. Sharan, R. Rosenholtz, and E. Adelson, "Material perception: What can you see in a brief glance?" *Journal of Vision*, vol. 9, no. 8, pp. 784–784, 2009.

[37] D. Ujević, S. Kovačević, L. Wadsworth, I. Schwarz, and B. B. Šajatović, "Analysis of artificial leather with textile fabric on the backside," *Journal of Textile and Apparel, Technology and Management*, vol. 6, no. 2, 2009.

[38] H. Tamura, S. Mori, and T. Yamawaki, "Textural features corresponding to visual perception," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 8, no. 6, pp. 460–473, 1978.

[39] Y. Hu, Z. Long, and G. AlRegib, "A high-speed, real-time vision system for texture tracking and thread counting," *IEEE Signal Processing Letters*, vol. 25, no. 6, pp. 758–762, 2018.

[40] V. Ferrari and A. Zisserman, "Learning visual attributes," in *Advances in Neural Information Processing Systems*, 2008, pp. 433–440.

[41] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision*, 1999, p. 1033.

[42] R. M. Haralick, K. Shanmugam, I. Dinstein, *et al.*, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[43] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, no. 5802, p. 91, 1981.

[44] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[45] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikäinen, "From bow to cnn: Two decades of texture representation for texture classification," *International Journal of Computer Vision*, vol. 127, no. 1, pp. 74–109, 2019.

[46] K. J. Dana, B. G. Van, S. K. Nayar, and J. J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Transactions On Graphics*, vol. 18, no. 1, pp. 1–34, 1999.

[47] B. Caputo, E. Hayman, and P. Mallikarjuna, "Class-specific material categorisation," in *IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1597–1604.

[48] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Opensurfaces: A richly annotated catalog of surface appearance," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 111, 2013.

[49] M. Strese, J. Lee, C. Schuwerk, Q. Han, H. Kim, and E. Steinbach, "A haptic texture database for tool-mediated texture recognition and classification," in *IEEE International Symposium on Haptic, Audio and Visual Environments and Games Proceedings*, 2014, pp. 118–123.

[50] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3479–3487.

[51] T. Wang, J. Zhu, E. Hiroaki, M. Chandraker, A. A. Efros, and R. Ramamoorthi, "A 4d light-field dataset and cnn architectures for material recognition," in *European Conference on Computer Vision*, 2016, pp. 121–138.

[52] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[53] H. Bay, T. Tuytelaars, and V. L. Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, 2006, pp. 404–417.

[54] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision*, 2011, pp. 2548–2555.

[55] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311.

[56] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu, "Discovering texture regularity as a higher-order correspondence problem," in *European Conference on Computer Vision*, 2006, pp. 522–535.

[57] W. Lin and Y. Liu, "Tracking dynamic near-regular texture under occlusion and rapid movements," in *European Conference on Computer Vision*, 2006, pp. 44–55.

[58] M. Varma and A. Zisserman, "A statistical approach to material classification using image patch exemplars," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2032–2047, 2009.

[59] M. Harandi, M. Salzmann, and F. F. Porikli, "Bregman divergences for infinite dimensional covariance matrices," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1003–1010.

[60] Y. Hu, Z. Long, and G. AlRegib, "Completed local derivative pattern for rotation invariant texture classification," in *IEEE International Conference on Image Processing*, 2016, pp. 3548–3552.

[61] L. Liu, P. Fieguth, M. Pietikäinen, and S. Lao, "Median robust extended local binary pattern for texture classification," in *IEEE International Conference on Image Processing*, 2015, pp. 2319–2323.

[62] Z. Guo, X. Wang, J. Zhou, and J. You, "Robust texture image representation by scale selective local binary patterns," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 687–699, 2016.

[63] Y. Hu, Z. Long, and G. AlRegib, "Scale selective extended local binary pattern for texture classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 1413–1417.

[64] R. Mehta and K. Eguiazarian, "Texture classification using dense micro-block difference," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1604–1616, 2016.

[65] Y. Hu, Z. Wang, and G. AlRegib, "Texture classification using block intensity and gradient difference (bigd) descriptor," *submitted to Signal Processing and Image Communication*, 2019.

[66] Y. Hu, Z. Long, A. Sundaresan, M. Alfarraj, and G. AlRegib, "Assessment of deep learning-based texture attributes using a challenging dataset," *submitted to Transactions of Multimedia*, 2019.

[67] L. Liu, P. Fieguth, X. Wang, M. Pietikäinen, and D. Hu, "Evaluation of LBP and deep texture descriptors with a new robustness benchmark," in *European Conference on Computer Vision*, 2016, pp. 69–86.

[68] X. Yang and K. T. Cheng, "Local difference binary for ultrafast and distinctive feature description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 188–194, 2014.

[69] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, 2010.

[70] S. Liao, M. W. K. Law, and A. Chung, "Dominant local binary patterns for texture classification," *IEEE Transactions on Image Processing*, vol. 18, no. 5, pp. 1107–1118, 2009.

[71] B. Zhang, Y. Gao, S. Zhao, and J. Liu, "Local derivative pattern versus local binary pattern: Face recognition with high-order local pattern descriptor," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 533–544, 2010.

[72] L. Liu, L. Zhao, Y. Long, G. Kuang, and P. Fieguth, "Extended local binary patterns for texture classification," *Image and Vision Computing*, vol. 30, no. 2, pp. 86–99, 2012.

[73] Z. Guo, Q. Li, J. You, D. Zhang, and W. Liu, "Local directional derivative pattern for rotation invariant texture classification," *Neural Computing and Applications*, vol. 21, no. 8, pp. 1893–1904, 2012.

[74] L. Liu, S. Lao, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Median robust extended local binary pattern for texture classification," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1368–1381, 2016.

[75] Y. Zhao, D. Huang, and W. Jia, "Completed local binary count for rotation invariant texture classification," *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4492–4497, 2012.

[76] Q. Lin and W. Qi, "Multi-scale local binary patterns based on path integral for texture classification," in *IEEE International Conference on Image Processing*, 2015, pp. 26–30.

[77] X. Qi, Y. Qiao, C. Li, and J. Guo, "Multi-scale joint encoding of local binary patterns for texture and material classification," in *British Machine Vision Conference*, 2013.

[78] Z. Li, G. Liu, Y. Yang, and J. You, "Scale-and rotation-invariant local binary pattern using scale-adaptive texton and subuniform-based circular shift," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2130–2140, 2012.

[79] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh, "On the significance of real-world conditions for material classification," in *European Conference on Computer Vision*, 2004, pp. 253–266.

[80] W. Zhang, W. Zhang, K. Liu, and J. Gu, "A feature descriptor based on local normalized difference for real-world texture classification," *IEEE Transactions on Multimedia*, 2017.

[81] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[82] L. Liu, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Local binary features for texture classification: Taxonomy and experimental study," *Pattern Recognition*, vol. 62, pp. 135–160, 2017.

[83] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (MSER) tracking," in *Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 553–560.

[84] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[85] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1508–1515.

[86] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, Citeseer, vol. 15, 1988, p. 50.

[87] T. Ojala, T. Mäenpää, M. Pietikainen, J. Viertola, J. Kyllönen, and S. Huovinen, "Outex-new framework for empirical evaluation of texture analysis algorithms," in *Conference on Pattern Recognition*, vol. 1, 2002, pp. 701–706.

[88] X. Qi, R. Xiao, C. Li, Y. Qiao, J. Guo, and X. Tang, "Pairwise rotation invariant co-occurrence local binary pattern," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2199–2213, 2014.

[89] F. M. Khellah, "Texture classification using dominant neighborhood structure," *IEEE Transactions on Image Processing*, vol. 20, no. 11, pp. 3270–3279, 2011.

[90] N. Shadkam, M. S. Helfroush, and K. Kazemi, "Local binary patterns partitioning for rotation invariant texture classification," in *IEEE CSI International Symposium on Artificial Intelligence and Signal Processing*, 2012, pp. 386–391.

[91] A. Fathi and A. R. Naghsh-Nilchi, "Noise tolerant local binary pattern operator for efficient texture analysis," *Pattern Recognition Letters*, vol. 33, no. 9, pp. 1093–1100, 2012.

[92] L. Liu and P. Fieguth, "Texture classification from random features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 574–586, 2012.

[93] Y. Hu, Z. Long, and G. AlRegib, "Scale selective extended local binary pattern for texture classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 1413–1417.

[94] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision*, 2010, pp. 143–156.

[95] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013.

[96] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *European Conference on Computer Vision*, 2010, pp. 778–792.

[97] P. Mallikarjuna, M. Fritz, A. Targhi, E. Hayman, B. Caputo, and J. Eklundh, *The KTH-TIPS and KTH-TIPS2 databases*, 2006.

[98] A. Vedaldi and B. Fulkerson, "VLFeat - An open and portable library of computer vision algorithms," in *Proceedings of the 18th ACM International Conference on Multimedia*, 2010, pp. 1469–1472.

[99] K. Valkealahti and E. Oja, "Reduced multidimensional co-occurrence histograms in texture classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 90–94, 1998.

[100] S. U. Hussain and B. Triggs, "Visual recognition using local quantized patterns," in *European Conference on Computer Vision*, 2012, pp. 716–729.

[101] G. Sharma, S. U. Hussain, and F. Jurie, "Local higher-order statistics (LHS) for texture categorization and facial analysis," in *European Conference on Computer Vision*, Springer, 2012, pp. 1–12.

[102] M. Crosier and L. Griffin, "Using basic image features for texture classification," *International Journal of Computer Vision*, vol. 88, no. 3, pp. 447–460, 2010.

[103] X. Hong, G. Zhao, M. Pietikäinen, and X. Chen, "Combining lbp difference and feature correlation for texture description," *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2557–2568, 2014.

[104] J. Ryu, S. Hong, and H. S. Yang, "Sorted consecutive local binary pattern for texture classification," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2254–2265, 2015.

[105] T. Nguyen, N. Vu, and A. Manzanera, "Statistical binary patterns for rotational invariant texture classification," *Neurocomputing*, vol. 173, pp. 1565–1577, 2016.

[106] R. Wang, H. Guo, L. Davis, and Q. Dai, "Covariance discriminative learning: A natural and efficient approach to image set classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2496–2503.

[107] R. Timofte and J. L. G. Van, "A training-free classification framework for textures, writers, and materials," in *British Machine Vision Conference*, vol. 13, 2012, p. 14.

[108] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, 2014, pp. 647–655.

[109] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural Computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[110] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[111] W. T. Freeman and J. B. Tenenbaum, "Learning bilinear models for two-factor problems in vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 554–560.

[112] J. Xue, *Deep-Encoding-Pooling-Network-DEP-*, `https://github.com/jiaxue1993/Deep-Encoding-Pooling-Network-DEP-.git/`, [Online; accessed 04-Aug-2018], 2018.

[113] Y. Hu, Z. Long, and G. AlRegib, "Multi-level texture encoding and representation (multer) based on deep neural networks," in *IEEE International Conference on Image Processing*, 2019.

[114] Y. Zhai, D. L. Neuhoff, and T. N. Pappas, "Local radius index-a new texture similarity feature," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1434–1438.

[115] Y. Hu, Z. Long, A. Sundaresan, M. Alfarraj, and G. AlRegib, "Assessment of deep learning-based texture representations using a challenging dataset," *submitted to IEEE Transactions on Multimedia*, 2019.

VITA

Yuting Hu received her bachelor's degree in electronic science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2011. In 2014, she received her master's degrees in electronic and communication engineering from Shanghai Jiao Tong University and electrical and computer engineering (ECE) from Georgia Tech, Atlanta, GA, USA. She is currently a graduate research assistant working on her doctoral degree in ECE at Georgia Tech. As a Ph.D. student of Dr. Ghassan AlRegib, she is an active member in the Omni lab for intelligent visual engineering and science (OLIVES). Her research interests lie in image and video processing, computer vision, machine learning, and texture analysis.